SMC-TR-93-13

AD-A262 212

AEROSPACE REPORT NO.
TR-93(3411)-2

# Minimizing Error in Predicting the Reliability of On-Board Satellite Systems

Prepared by

D. A. LEE, JR

Edited by

G. C. GILLEY
Computer Systems Division
Engineering and Technology Group

DTIC
ELECTE
MAR 25 1993
S E D

October 1992

Prepared for

SPACE AND MISSILE SYSTEMS CENTER
AIR FORCE MATERIEL COMMAND
Los Angeles Air Force Base
P. O. Box 92960
Los Angeles, CA 90009-2960

Programs Group

THE AEROSPACE CORPORATION
El Segundo, California

APPROVED FOR PUBLIC RELEASE;
DISTRIBUTION UNLIMITED

93-06092

93 3 24 032

This final report was submitted by The Aerospace Corporation, El Segundo, CA 90245-4691, under Contract No. F04701-88-C-0089 with the Space Systems Division, P. O. Box 92960, Los Angeles, CA 90009-2960. It was reviewed and approved for The Aerospace Corporation by H. J. Wertz, Principal Director, Computer Engineering Subdivision, Computer Systems Division, Engineering and Technology Group and J. R. Parsons, Principal Director, Follow-on Early Warning System, Space Program Operations, Programs Group. The project officer is Mario Miranda, SSD/MBT.

This report has been reviewed by the Public Affairs Office (PAS) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nationals.

This technical report has been reviewed and is approved for publication. Publication of this report does not constitute Air Force approval of the report's findings or conclusions. It is published only for the exchange and stimulation of ideas.

FOR THE COMMANDER

JOSEPH CHIARA
for M. Miranda

## REPORT DOCUMENTATION PAGE

| | |
|---|---|
| 1a. REPORT SECURITY CLASSIFICATION<br>Unclassified | 1b. RESTRICTIVE MARKINGS |
| 2a. SECURITY CLASSIFICATION AUTHORITY | 3. DISTRIBUTION/AVAILABILITY OF REPORT<br>Approved for public release; distribution unlimited |
| 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE | |
| 4. PERFORMING ORGANIZATION REPORT NUMBER(S)<br>TR-93(3411)-2 | 5. MONITORING ORGANIZATION REPORT NUMBER(S)<br>SMC-TR-93-13 |
| 6a. NAME OF PERFORMING ORGANIZATION<br>The Aerospace Corporation<br>6b. OFFICE SYMBOL *(If applicable)* | 7a. NAME OF MONITORING ORGANIZATION<br>Space and Missile Systems Center<br>Air Force Materiel Command |
| 6c. ADDRESS *(City, State, and ZIP Code)*<br>2350 E. El Segundo Blvd.<br>El Segundo, CA 90245-4691 | 7b. ADDRESS *(City, State, and ZIP Code)*<br>Los Angeles Air Force Base<br>P. O. Box 92960<br>Los Angeles, CA 90009-2960 |
| 8a. NAME OF FUNDING/SPONSORING ORGANIZATION<br>Space and Missile Systems Center<br>Air Force Materiel Command<br>8b. OFFICE SYMBOL *(If applicable)*<br>MB | 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER |
| 8c. ADDRESS *(City, State, and ZIP Code)*<br>Los Angeles Air Force Base<br>P. O. Box 92960<br>Los Angeles, CA 90009-2960 | 10. SOURCE OF FUNDING NUMBERS<br>PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO. |

11. TITLE *(Include Security Classification)*

Minimizing Error in Predicting the Reliability of On-Board Satellite Systems

12. PERSONAL AUTHOR(S)

D. A. Lee, Jr. and G. C. Gilley; Ed.

| 13a. TYPE OF REPORT | 13b. TIME COVERED | 14. DATE OF REPORT *(Year, Month, Day)* | 15. PAGE COUNT |
|---|---|---|---|
| Final | FROM 6/91 TO 6/92 | October 1992 | 72 |

16. SUPPLEMENTARY NOTATION

| 17. COSATI CODES | | | 18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)* |
|---|---|---|---|
| FIELD | GROUP | SUB-GROUP | |
| | | | |
| | | | |

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

The reliability prediction process for large, closed (i.e., non-repairable) fault-tolerant, on-board satellite systems is addressed, along with techniques for reduction of error in prediction. Three major error sources are discussed, only one of which can be effectively minimized by modelers. Specific error sources in model construction are identified. It is presented that, although errors exist in the reliability prediction process, complete accuracy is not required for effective use of the predictions in the design process. A set of three tools [(CARE III [Bavu 84a, Bavu 84b], HARP [HARP 89], and CRAFTS [CRAF 88]) are used to model an example on-board system design to illustrate minimization of errors. Guidelines for selecting the proper reliability tools are also given.

| 20. DISTRIBUTION/AVAILABILITY OF ABSTRACT | 21. ABSTRACT SECURITY CLASSIFICATION |
|---|---|
| [X] UNCLASSIFIED/UNLIMITED [X] SAME AS RPT. [ ] DTIC USERS | Unclassified |

| 22a. NAME OF RESPONSIBLE INDIVIDUAL | 22b. TELEPHONE *(Include Area Code)* | 22c. OFFICE SYMBOL |
|---|---|---|
| Mario Miranda | (310) 363-0958 | SMC/MBT |

# PREFACE

We wish to thank Joanne Dugan for explaining the operation of the new fault tree gates used in HARP.

Accesion For
NTIS CRA&I
DTIC TAB
Unannounced
Justification
By
Distribution /
Availability Codes
Dist | Avail and / or Special
A-1

DTIC QUALITY INSPECTED 1

CONTENTS

CONTENTS (Continued

## FIGURES



## TABLES

# 1 INTRODUCTION

## 1.1 PURPOSE

This report discusses how to reduce the error in predicting the reliability of large, closed (i.e., nonrepairable), fault-tolerant, on-board satellite systems, which will hereafter be referred to as on-board systems (OBSs). This report also describes the reliability prediction process itself in order to provide a better understanding of what the reliability prediction results actually mean and how to use the results more effectively in the OBS design process.

This report describes the three major sources of error that exist in the reliability prediction process: (1) model construction, (2) tool limitations, and (3) solution and representation errors. Only model construction errors can be effectively minimized by modelers because this is the only error source over which they have control. Specific error sources in model construction are identified in this report, so modelers can avoid these errors in constructing models.

Although errors exist in the reliability prediction process, the results are not useless. Depending upon how the predictions are used, total accuracy in the process is not necessary to effectively use the predictions in the design process. For example, reliability predictions can be used to select the correct preliminary design approach from the initial set of design alternatives despite inaccuracies in the predictions.

A set of three tools (CARE III [Bavu 84a, Bavu 84b], HARP [HARP 89], and CRAFTS [CRAF 88]) are used in this report to correctly model an example OBS design to illustrate how to minimize the introduction of errors. Guidelines on model construction are given to reduce the variability in the OBS model and the model's results, so design evaluators can better understand and use the results from different modeling efforts.

This report also presents guidelines for selecting the correct reliability prediction tools. Modelers should avoid using tools with limitations preventing the construction of accurate models. Tool selection involves comparing each tool's ability to accurately model four design features: (1) design structure, (2) fault arrival process, (3) fault/error handling, and (4) coverage treatment.

Finally, recommendations are given on the following topics:

a. How to modify MIL-STD 756B to allow more accurate OBS modeling.
b. How to enhance the reliability prediction capability of system developers.

c. How to use a set of questions as a qualitative tool to improve the reliability prediction process.
d. How to improve reliability prediction tools.

Funding for the work addressed by this report was terminated prior to the completion of the total proposed effort. The early termination has not impacted the accuracy or the scope of the material presented in this report, but only the level of detail in the modeling examples and descriptions of tool features.

## 1.2 THE RELIABILITY PREDICTION PROCESS

The process for predicting the reliability of OBSs is illustrated in Figure 1. The steps of the process are numbered, and the error sources are shown in parentheses. The three parts of this process are a design, a design model, and a reliability prediction tool. A design is a scheme to implement a set of algorithms in hardware and software. A design model is an abstraction of the design features that affect reliability. A design model provides a convenient way for modelers to represent these design features and manipulate them so predictions and investigations can be made of a design's reliability. In this report, the model includes all parameters provided to a specific reliability prediction tool (e.g., failure rates). A reliability prediction tool is a program used by modelers to manipulate the design model to obtain reliability predictions. The tool converts the design model into a set of mathematical equations and solves those equations.

Evaluators are people who use the predictions in both the design and the evaluation process. Evaluators can be modelers, designers, or independent design reviewers. In the design process, the interpretations of the predictions provide feedback to design evaluators for use in performing design tradeoffs, such as selecting a preliminary design approach and determining what redundancy techniques should be used at each design level (e.g., module, submodule, or chip). In the evaluation process, the predictions are interpreted by evaluators to determine if the chosen design satisfies the reliability requirements.

Reliability prediction is performed repeatedly throughout the design process. As the design evolves, the reliability prediction process addresses ever increasing design detail and attempts to produce increasingly accurate results. Early in the design cycle, a design model is constructed from preliminary design information and rough estimates of design parameters (e.g., coverages). Consequently, the results may not accurately predict the reliability of the final design. However, as the design becomes better defined and more accurate estimates of parameter values become available, repeating the reliability prediction process produces predictions that are

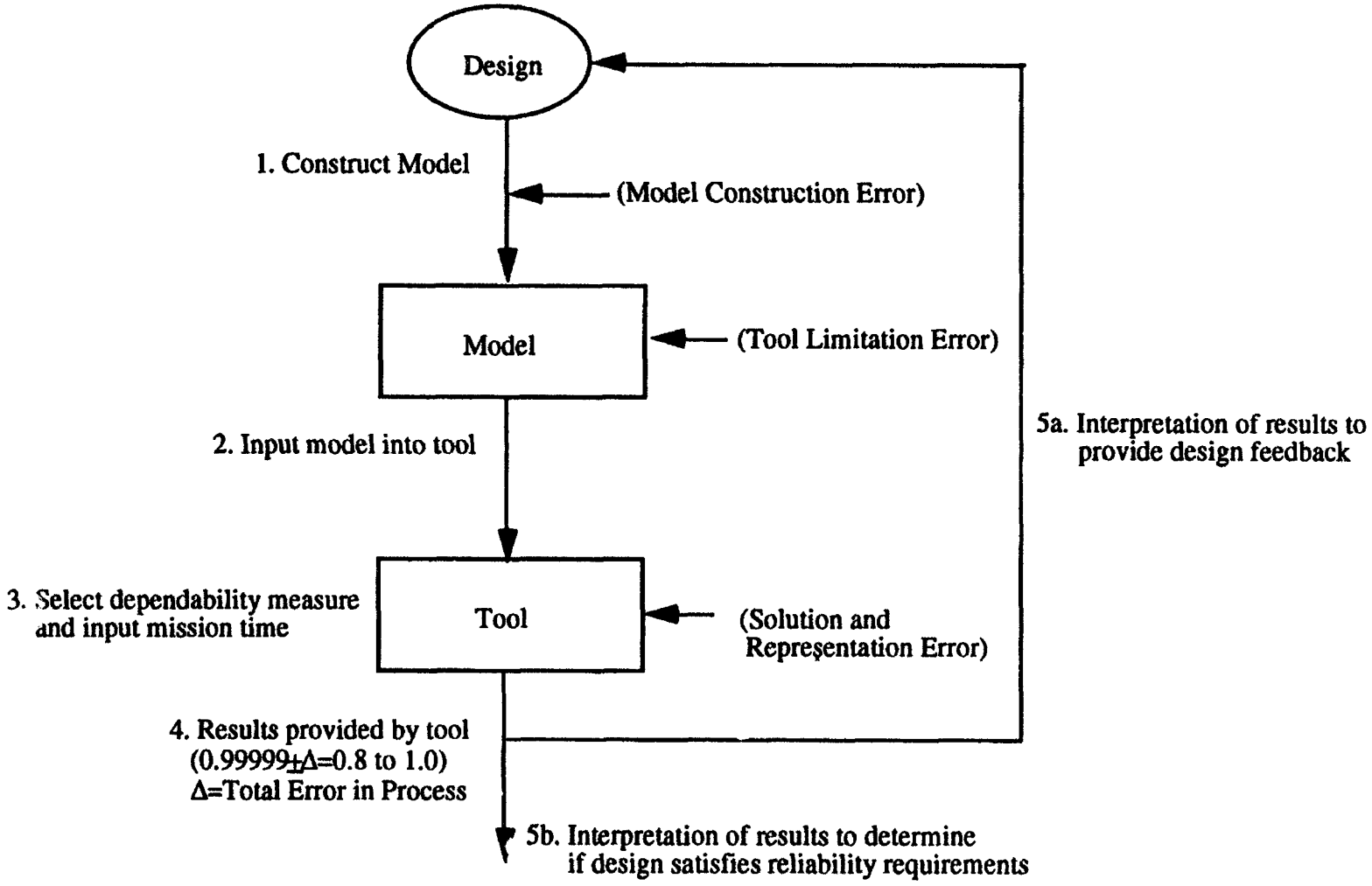



Figure 1. Reliability Prediction Process

increasingly more accurate indicators of a design's reliability. As a result, as the design evolves, modelers must continue to refine existing design models by including more accurate estimates of design parameters as the estimates become available or construct new design models to reflect the design's current state.

## 1.3 ERRORS IN THE PROCESS

The major categories of error shown in Figure 1 (in parentheses) are: model construction errors, solution and representation errors, and tool limitation errors. As a result of these errors, the predictions provided by the tool may not be accurate even in the first decimal place. The following paragraphs describe each error category.

Model construction errors occur when mistakes are made in repre-senting a design in a model. For example, a model construction error occurs when a modeler represents system components as operating in parallel when they actually operate in series.

Tool limitation errors are caused by a tool's inability to represent design features that impact reliability. As a result, tool limitations prevent modelers from accurately representing those features in a design model. For example, the reliability prediction tool CARE III allows the modeling of hot (i.e., active) spares but not cold (i.e., unpowered) spares.

Approximations in solving the model or in representing results cause solution and representation errors. For example, a tool's solution technique may introduce inaccuracy into a prediction as a result of truncation. Truncation occurs when a continuous derivative is discretized or a series expansion is used for various functions.

## 1.4 REQUIRED TOOL CAPABILITIES

The author believes that a reliability prediction tool should have, as a minimum, the capability to model the following seven important OBS design features that impact reliability, in order to provide accurate feedback to designers performing tradeoffs or evaluations:

a. redundancy
b. sparing
c. dependencies
d. permanent, transient, and near-coincident faults
e. fault distributions

f. fault/error handling
g. hierarchical coverage treatment

### 1.4.1 Redundancy

A tool should provide the ability to model the different redundancy strategies used in an OBS. For example, system developers may use static, dynamic, or hybrid redundancy to meet the reliability requirements.

### 1.4.2 Sparing

A tool should accurately model the three sparing strategies used in OBS designs: hot, warm, and cold. Hot spares are "powered-up" standby elements that contain the current system state and can replace failed elements immediately. Warm spares are partially "powered-up." Warm spares use less power either by not driving a clock or applying full power to their circuits. Cold spares are "powered-down." Warm and cold spares cannot immediately replace a failed element, since they must be brought up to full power and "loaded" with the appropriate information.

### 1.4.3 Dependencies

Reliability prediction tools should provide the ability to accurately model the functional and sequence dependencies that can exist in an OBS design. A functional dependency occurs when a component failure makes another component unavailable. Figure 2 illustrates a functional dependency where the failure of component C makes components A and B unavailable.

There are two types of sequence dependencies. The first type occurs when component failures must follow a certain sequence for the system to fail. The second type occurs when the actions that a system undertakes depend upon what component failures have already occurred.

Figure 3 illustrates a system with the first type of sequence dependency. The system consists of two sets of computers (A and B). Each set contains two computers. Initially, assume both computers in set A are powered ON and are operating, and the computers in set B are powered OFF. Upon the failure of both computers in set A, set B's computers are powered ON and become operational. Upon the failure of both computers in set B, the system fails. The sequence dependency aspect here is that set B is not even considered until after set A fails.

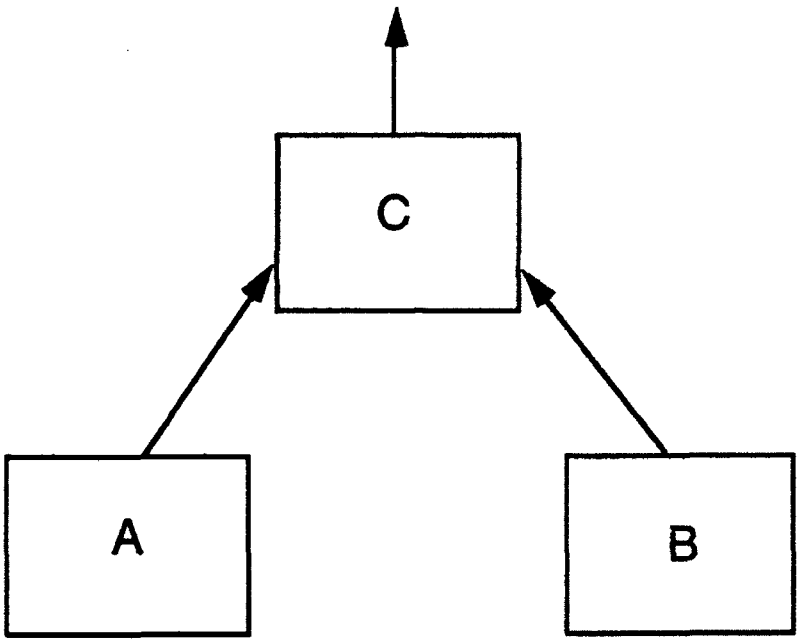



Figure 2. A Functional Dependency

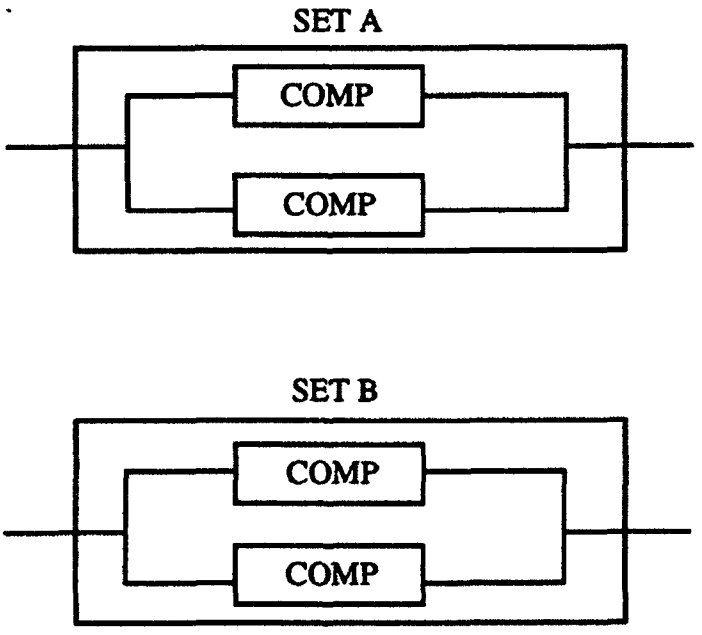



Figure 3. Type One Sequence Dependency

Figure 4 illustrates the second type of dependency using a two-channel system. Each channel consists of two sensors connected to a CPU through a device interface unit (DIU). The sensors in each channel operate in parallel. A channel fails if both sensors or the CPU or the DIU fails. The system requires only one of the two channels to be operational.

At first, assume the system uses only channel 1. The two-channel system can take one of two actions upon a channel 1 component failure. First, the system immediately switches to channel 2, if channel 1's CPU or DIU fails and channel 2 has suffered only a sensor failure. Subsequently, the system continues to operate using channel 2 until it fails. At this point, the system fails. Second, if a sensor fails in channel 1 and channel 2's CPU or DIU has failed before the channel 1 sensor, the system can continue to operate using channel 1 until the next component failure. At that time, the system fails.

### 1.4.4 Fault Types

A reliability prediction tool should be able to correctly model three types of faults that can affect OBSs: single permanent, single transient, and near-coincident. A near-coincident fault is a fault that occurs while a system is still attempting recovery from a previous fault.

### 1.4.5 Fault Distributions

To model the arrival of these three fault types, a tool should provide the exponential and Weibull distributions. Generally, exponential distributions are used to model fault arrival in electronic components, and Weibull distributions are used to model fault arrival in nonelectronic components. For the distribution associated with each component, a rate is required that describes the frequency of fault arrival over time.

### 1.4.6 Fault/Error Handling

A reliability prediction tool should accurately model the steps an OBS follows in detecting, isolating, and reconfiguring from faults and their associated errors. Additionally, the physical aspects of fault behavior should be included in the model. The modeling of fault and error handling steps allows a modeler to perform parametric studies to determine how varying the effectiveness of these steps impacts OBS reliability.

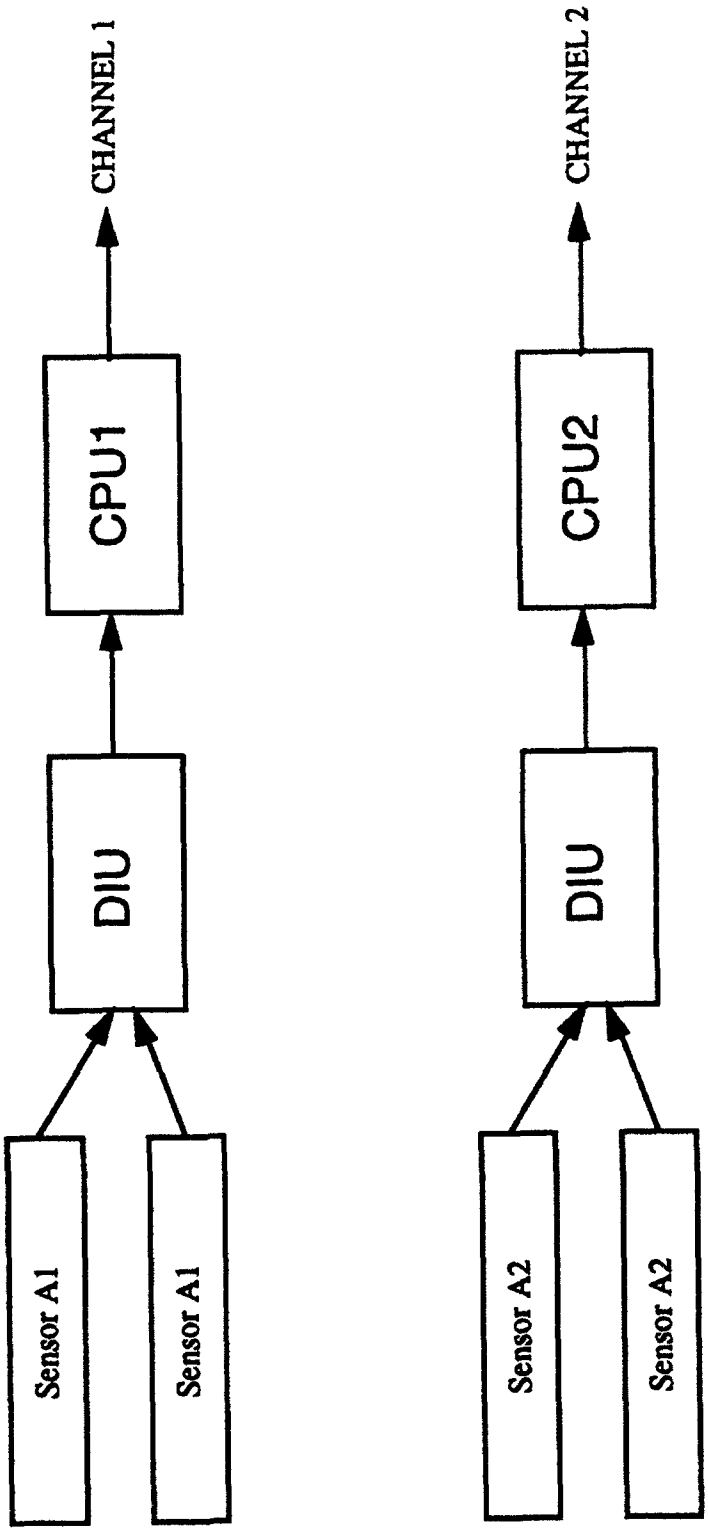



Figure 4. Type Two Sequence Dependency

### 1.4.7 Hierarchical Coverage Treatment

A reliability prediction tool should allow coverage values to be included in an OBS model at any design level. OBS designs typically incorporate a hierarchical treatment of and recovery from errors. All errors are not necessarily treated the same way at each hierarchy level. Coverage is a measure of how effectively a design detects and recovers from errors at a particular hierarchy level. If the coverages associated with each hierarchy level are not accurately included in an OBS model, the reliability prediction feedback to designers may result in more costly and less adequate designs.

### 1.4.8 Modeling Techniques

This section describes the three modeling techniques used in the reliability prediction tools commonly used by system developers: fault trees, reliability block diagrams, and Markov models. The shortcomings of these techniques in accurately modeling OBSs are described. Other reliability prediction techniques, such as Petri Nets and Monte Carlo simulation, will not be described, since they are not as commonly used by OBS developers.

### 1.4.9 Fault Trees

A fault tree is a graphical representation of the combinations of events that lead to system failure. The conditions of system failure are decomposed by the use of logic gates (AND, OR, and M-of-N gates) until basic events are reached. AND-gates represent the parallel operation of system components, OR-gates represent the series operation of system components, and M-of-N gates represent the failure of m-out-of-n components. A basic event represents the failure of a component at the lowest design level (e.g., the failure of individual CPUs, memories, or buses).

### 1.4.10 Reliability Block Diagrams

Reliability block diagrams (RBDs) reflect the reliability behavior of a design by connecting design components in series-parallel paths. In reliability block diagrams, design failure can be thought of as the inability to trace a path from a source at one end of the diagram to a sink at the opposite end of the diagram due to the removal of components by failures.

Fault trees and RBDs cannot accurately represent four important OBS design features in a model. First, warm and cold spares are treated as active components which prevents the use of

accurate failure rates for the spares and creates artificial situations where transients may cause the loss of spares. Fault trees and RBDs represent hot spares correctly. Hot spares are treated the same as active components because their failure rates are the same, and transients must be given the same consideration whether the component is active or a hot spare. Second, functional and sequence dependencies cannot be modeled. Therefore, the model's results are not indicative of a design's actual reliability. The amount of error introduced cannot be determined, since the error is a function of the number of dependencies not modeled, the failure rates of the components in each dependency, the redundancy strategies used with the components, and the coverage values of the fault tolerance mechanisms. Third, coverage values currently cannot be included in an OBS model at every design level. As a result, the models provide optimistic predictions. Fourth, fault and error handling cannot be modeled. This prevents modelers from obtaining feedback on how variations in the steps a design takes in handling faults and their resulting errors impacts OBS reliability.

Recently, the latest version of HARP, version 6, introduced new fault tree gates that overcome fault tree's inability to accurately represent dependencies and any sparing strategy. The new gate types are available only with this version of HARP and will be discussed in the section on HARP.

A Markov model represents a design as a set of states and transitions between those states. A state represents a unique combination of correctly operating and failed design components. A transition represents the occurrence of faults, fault and error handling, or the failure of a design component. Generally, Markov models can correctly address all seven design features listed at the beginning of Section 1.4. Markov models allow accurate design models to be constructed, since modelers can use Markov models to represent each specific system state and the specific causes of transition between those states in a model (e.g., failure of a warm spare).

There are two problems with using Markov models to predict the reli-ability of OBSs. The first problem is analyzing large state spaces. In Markov models, each combination of correctly operating and failed design components must be represented by a state. Since the number of states can increase ex-ponentially with the number of components modeled, the models can be so large that they exceed the capability of the reliability prediction tool or the capacity of the computing resource. The second problem is that the construc tion of Markov models is a time-consuming, error-prone task in all but the simplest cases.

Workarounds exist for the problems with Markov models. For the first problem, several mathematical approaches have been devised to aid in analyz-ing Markov models with large state spaces. One technique, behavioral decom-position, separates the Markov model according to the relative magnitude of the state-transition rates. All fast transitions are solved separately and the results are used to modify the transition rates of the remaining model. Behavioral decomposition is the approach used in CARE III and HARP. Another technique is the use of a mixture of modeling techniques. For example, CRAFTS uses Markov models to solve subsystem reliability, and the results of these Markov models are used as inputs to a combinatorial solution of the overall system reliability. Thus, the problem of analyzing a large state space model is avoided.

To workaround the time-consuming, error-prone task of constructing Markov models, a more succinct form of system description is used by the tools to generate the Markov model. CARE III and HARP use fault trees to succinctly describe the system design, and CRAFTS uses reliability block diagrams.

## 2. ACCURATE MODELS AND AVOIDANCE OF ERROR SOURCES

It is important to realize that a tool predicts the reliability of a model and not the reliability of a design. The reliability of a model corresponds only to the reliability of a design to the degree that the model actually represents a design. The greater the inaccuracy in a model, the less the reliability numbers produced by a tool reflect a design's actual reliability. Tool limitations can a priori limit the accuracy of model construction by not allowing the accurate representation of OBS design features, especially fault tolerance functions.

Of the three error sources described above, modelers can have the greatest impact on model construction errors because they have the greatest influence on this source of error. Modelers can do little to minimize error caused by a tool's limitations or a tool's solution technique. They generally have no control over these tool features. However, modelers can minimize error in model construction by ensuring the most accurate model possible is constructed.

Several error sources cause a design to be inaccurately represented in a model. By understanding these error sources, modelers can minimize errors and construct more accurate models. The error sources in model construction are: modeler assumptions, modeler misunderstanding, incorrect parameter value determination, and incorrect measurement. The following paragraphs describe each error source.

A modeler's assumptions about a design may result in an incorrect model. A modeler may make assumptions to simplify a model to make it tractable, or because the modeler feels that certain design features do not significantly impact design reliability [Triv 84]. However, the assumptions may be invalid. For example, a modeler may assume the mechanisms used to implement the fault-tolerance functions are themselves fault tolerant or the occurrence of faults follows a Poisson process. If these assumptions are invalid, the resulting model is incorrect.

If a modeler does not understand how a design operates, incorrect models can result. For example, a modeler may assume a subsystem fails when both of its components fail; whereas in actual practice, the subsystem might fail when either one of the components fails.

Incorrect models result when a modeler does not understand how to use correctly the model construction techniques that a tool provides. For example, a modeler may construct an incorrect fault tree model of a design because he or she did not understand how to use correctly a particular type of fault tree gate.

Errors in model parameter values are caused by incorrect system simulations, emulations, or incorrect analysis of experimental data. Erroneous parameter values can also result from incorrect calculations. For example, a modeler may use incorrect values while calculating component failure rates using MIL-HDBK 217 [MIL 86].

Mistakes in measurement also introduce error into parameter values. Measurement errors result from imprecision in collecting experimental data. Sources of measurement error are the physical limitations of measuring equip-ment and mistakes in recording or reporting data. Measurement errors must be especially minimized when experimental results are used as parameter values in the final design model.

# 3. EXAMPLE ON-BOARD SYSTEM

An example OBS is described here and used in the following section to illustrate how design models should be constructed. A block diagram of the example system is shown in Figure 5. The system is composed of the following subsystems: clock, power supply, system I/O (i.e., the I/O subsystem used by the node to communicate with other spacecraft systems), nodal I/O (i.e., the I/O subsystem used by the node to communicate with other identical nodes), array processor, scalar processor, and memory. The example system is considered failed if any one of its subsystems fails. To illustrate tool limitations, all subsystems are arbitrarily specified to use warm spares rather than hot or cold spares. The following paragraphs describe the subsystems in greater detail.

The clock, power supply, system I/O, nodal I/O, and array processor subsystems consist of one active component and one spare. At least one component of each subsystem must work for the node to be operational.

The scalar processor subsystem consists of three self-checking CPUs and one spare. At least two of the CPUs must operate.

The memory system consists of two component types: non-volatile memory (NVM) and random access memory (RAM). The NVM consists of three memory modules and a spare. Three of the NVM memory modules must work. The RAM consists of four memory modules and two spares. Four of the RAM memory modules must work.

Failure rates of subsystem components will be presented later in the report. The spares in each subsystem replace active components that have failed. Spares that have failed are unable to replace failed active components.

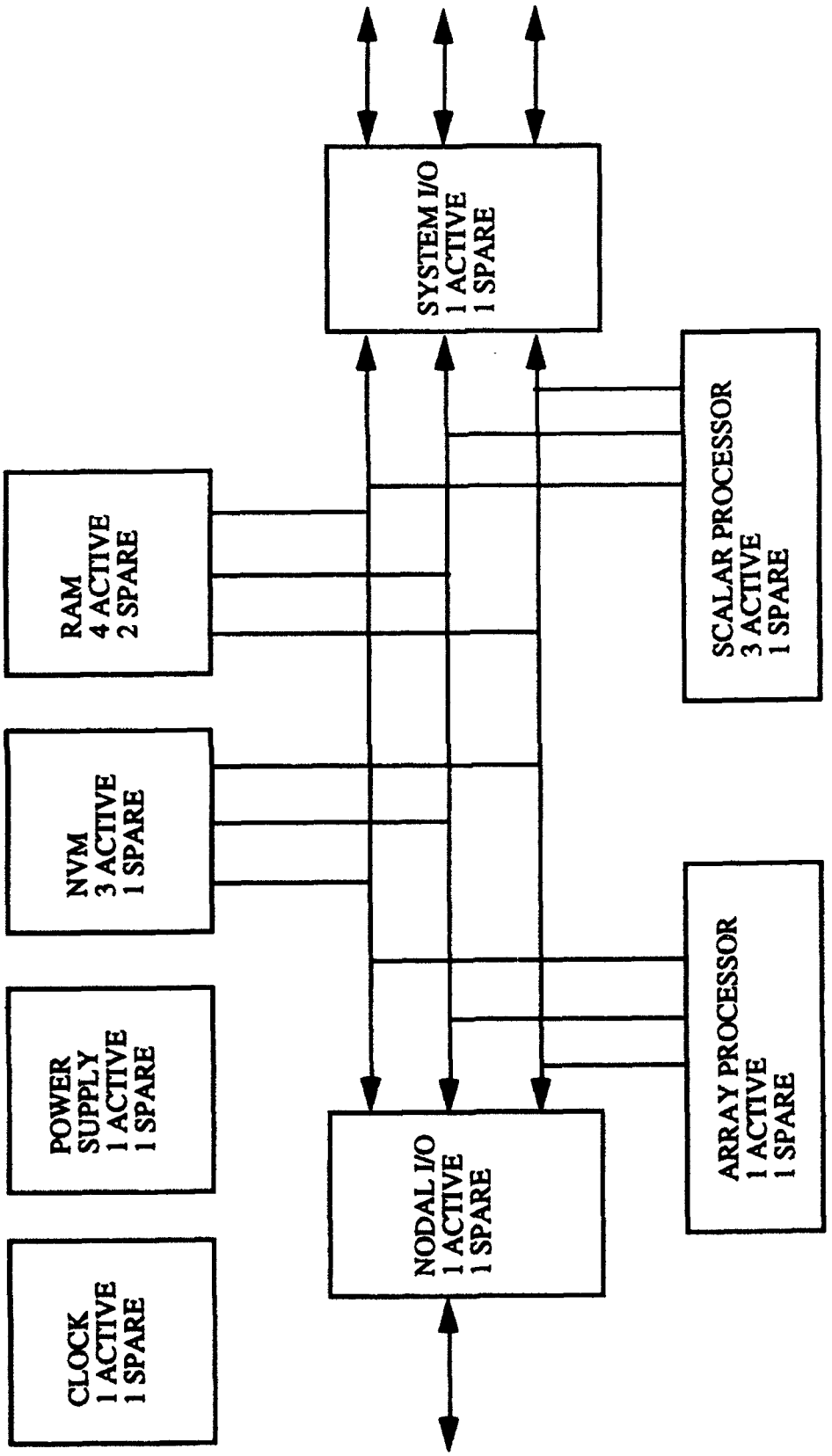



Figure 5. Example System Node

# 4 PERFORMING RELIABILITY PREDICTION

This section shows how to minimize errors effectively while performing reliability prediction. How design models should be constructed, how tools should be selected, and how tools should be used is shown. This is illustrated by showing how to predict the reliability of the example system using a fixed set of tools: CARE III, HARP, and CRAFTS.

## 4.1 CONSTRUCTING A CARE III MODEL

In this section, how to construct a CARE III model of our example system is discussed. CARE III uses Markov models to predict the reliability of OBSs.

CARE III does not require the inputs to the tool to actually be in the form of Markov models because the construction of Markov models is a time-consuming and error-prone task for all but the simplest cases. CARE III allows modelers to input fault trees plus additional information about fault arrival rates and fault/error handling. The tool then constructs and solves a Markov model representation of this information. To construct a CARE III model we must:

1. Build a fault tree of the system design.
2. Determine the failure rates of the lowest level components.
3. Determine the parameter values to be used in the Fault/Error Handling Model (FEHM) of each lower level component.

Since no design detail is given regarding the handling of faults and errors in the example design, using FEHM parameter values derived from simulations and emulations of the design detail is not discussed at this time. Instead, for each tool, the discussion focuses on the limitations of the tools' FEHM(s) and how to use those FEHMs more effectively.

### 4.1.1 Fault and Error Handling Models

In this section, a general discussion of FEHMs is provided before continuing the discussion of CARE III, so the reader can understand how the CARE III FEHM, and the FEHMs of other tools, relate to reliability prediction. FEHMs attempt to represent the behavior of a system in handling faults and their resulting errors in a model. The following paragraphs explain why FEHMs are used to represent a system's fault and error handling.

Figure 6 presents a simple Markov model of a system with two identical components and a total fault rate (i.e., the sum of the permanent and transient fault rates) of $\lambda$. The modeler inputs the permanent and transient fault rates into the tool. The model shows that the failed state "F" is reached only after both components have failed. The model also shows that after the first failure, the system will recover and continue to operate until the second failure occurs. However, this model is too simplistic to provide a detailed analysis of the effects of coverage on the detection and recovery mechanisms and the occurrence of other fault types (e.g., transient and near-coincident).

An artifice called an FEHM solves these problems. An FEHM models the steps a design follows in detecting, isolating, reconfiguring, and recovering from faults and their associated errors. In addition, FEHMs attempt to include the physical aspects of fault behavior (e.g., fault duration).

A general structure of an FEHM is shown in Figure 7. An FEHM has a single entry point, the occurrence of a fault, and up to four exits depending on the fault types modeled. For example, if transient faults are not modeled, the FEHM has no an exit for transient restoration.

Each exit represents a possible outcome of a design's fault handling. The C exit represents the probability of a design successfully handling a perma-nent, intermittent, or transient fault that is mistaken as permanent. The S exit represents the probability of a design failing due to unsuccessfully handling a single fault. The R exit represents the probability of a design correctly recover-ing from a transient fault. The N exit represents the probability of a design failing due to the occurrence of another fault that interferes with a design's recovery from a previous fault (i.e., a near-coincident fault). For the tool to determine this probability, a modeler must input the near-coincident fault rate.

In the case of these three tools, the near-coincident fault rate is defined as the total fault rate of the components performing the fault component's recovery. However, system designers consider any second fault during a recovery period as near-coincident. Therefore, the tools take a more narrow definition of near-coincident faults than do system designers.

Actual system designs contain mechanisms allowing recovery from near-coincident faults. However, the current tools consider the occurrence of a near-coincident fault as a system failure. As a result, modelers cannot model a system's actual handling of near-coincident faults.

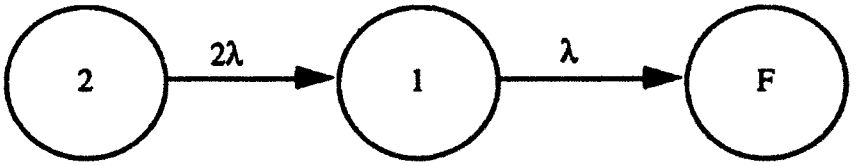



Figure 6. A Markov Model of a Two Component System

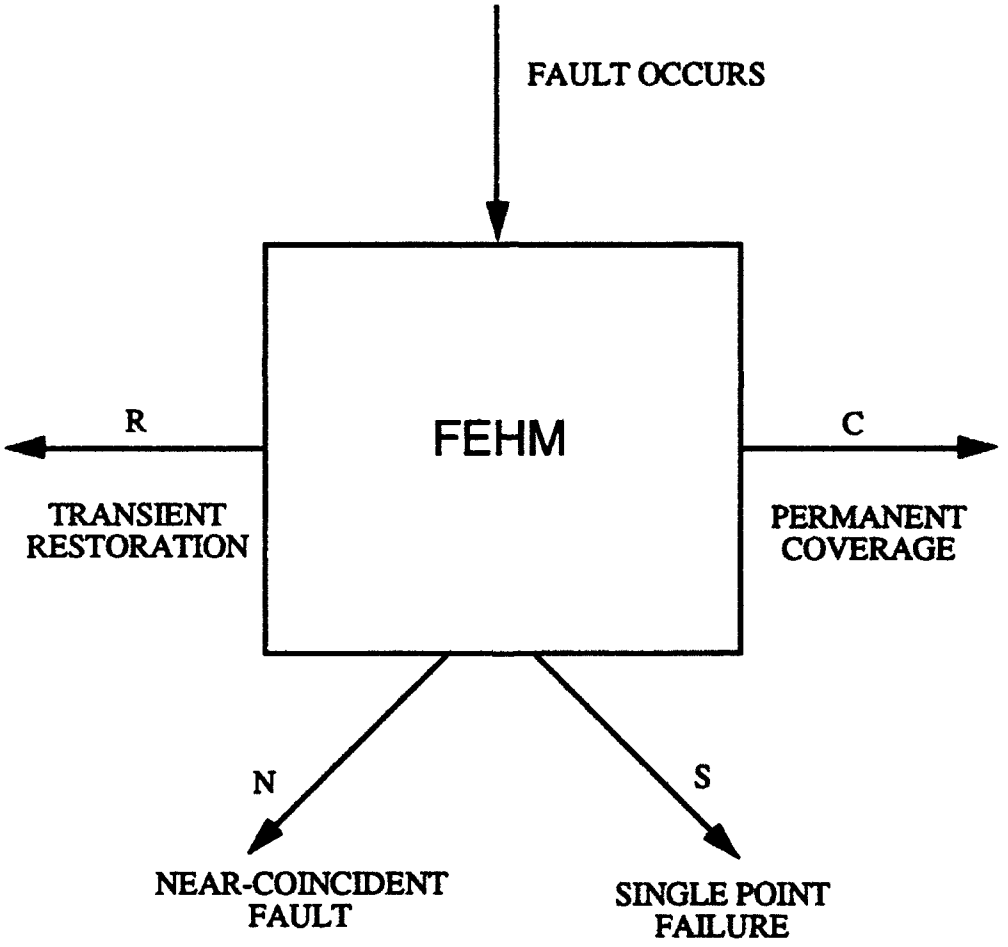



Figure 7. A General Description of an FEHM

Since the tools treat the occurrence of a near-coincident fault as a system failure, the tools' results are conservative.

Figure 8 shows our example model with an FEHM inserted in the model. Since a transition to another state does not occur when a design recovers from a transient fault, the R exit of the FEHM returns to state 2. The S and N exits each go to a new failed state. The SPF state represents design failure due to unsuccessful fault handling (other than near-coincident faults), and the NCF state represents design failure due to the occurrence of a near-coincident fault. The original failed state "F" now represents design failure due to resource exhaustion (i.e., both components fail). These three failure states provide feedback to a modeler about the specific causes of design failure.

Based on the parameters supplied by a modeler, the tool solves the FEHM to determine the transition rates for each exit of the FEHM together with the probability (coverage) of taking each exit. This process equates to statistically determining how all the transitions out of a given state are distributed to all other states.

In solving the Markov model, the tool first solves all the FEHMs. The tool then replaces the FEHMs with the resulting probabilities for the various transitions between states (i.e., the probabilities for the various transitions out of each state). Figure 9 shows the model of our example system after the FEHM solution.

### 4.1.2 Building the Fault Tree of the Example System

CARE III requires the use of fault trees to describe the conditions of system failure. To create a CARE III fault tree, the conditions of failure for each design level are iteratively defined, from the top level (i.e., system level) down to the bottom level (i.e., stage failures). For example, first the conditions of system failure are defined. Next, the conditions of subsystem failure are defined. Then, the conditions of failure for each level continue to be defined until the stage failures are reached.

A stage is a basic event in CARE III. A stage is a group of n identical components, at the lowest level of the design. A stage fails when m of the n components in a stage fail. The aforementioned logic gates are used to represent these conditions. To illustrate this procedure, the following paragraphs explain how to build the CARE III fault tree of the example system.

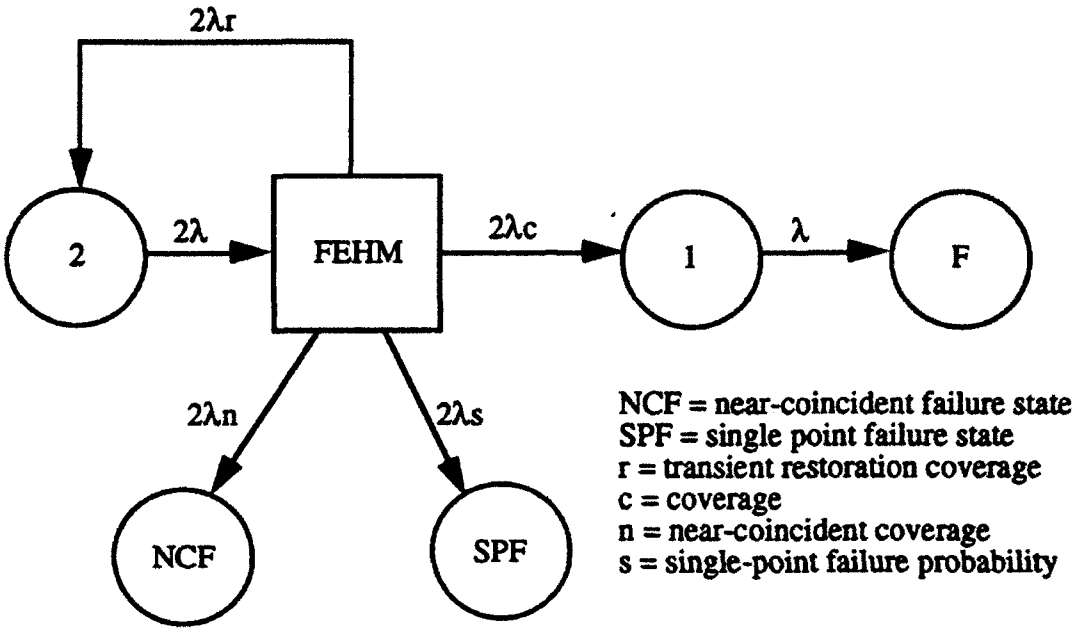



Figure 8. Markov Model of Two Component System with FEHM Inserted

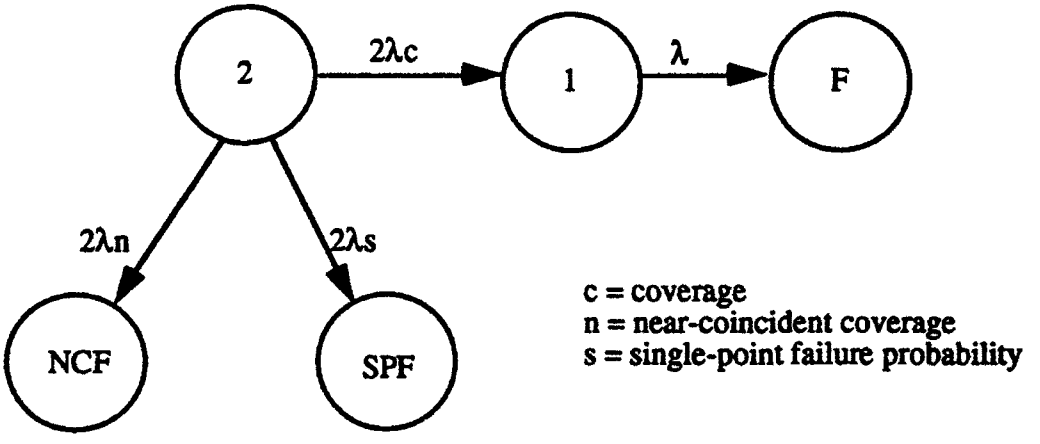



Figure 9. Markov Model After FEHM Solution

Figure 10 shows the completed fault tree of the example system. The event labeled "system failure" occurs if any subsystem fails. That is, the system fails, if the clock OR the power supply OR the nodal I/O OR ··· the RAM fails. Since the details of the example system design is given only down to the subsys-tem level, that level becomes the lowest level of the fault tree (i.e., stages).

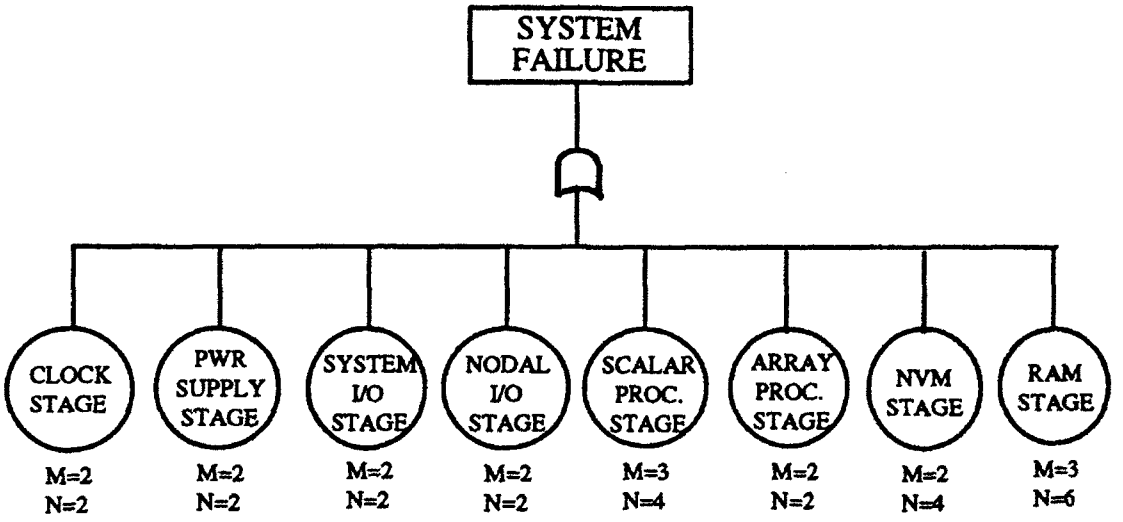



Figure 10. CARE III Fault Tree of Example System

Next, the conditions of failure for each stage are defined. That is, values are assigned to M and N for each stage. The clock, power supply, system I/O, nodal I/O, and array processor each fails, if both its active and spare components fail. Therefore, for these stages M=2 and N=2. The scalar processor subsystem fails if three out of the four processors fail, so for the scalar processor stage M=3 and N=4. The NVM subsystem fails if two of the four non-volatile memories fail, and the RAM subsystem fails if three of the six random access memories fail. Thus, M=2 and N=4 for the NVM stage, and M=3 and N=6 for the RAM stage.

In the CARE III fault tree, all the spares are considered hot, even though the example system uses warm spares. Thus, the CARE III results of the example system provide reduced accuracy, since the spares cannot be correctly modeled. CARE III's treatment of spares illustrates how a tool can prevent modelers from representing accurately a design feature that impacts reliability.

It is important to note that even though CARE III solves a Markov model, CARE III cannot create an accurate Markov model to represent depend-encies as well as warm and cold spares, since the fault trees cannot represent this information.

CARE III overcomes the inability of fault trees to represent the fault and error handling details of a design by using a Fault/Error Handling Model (FEHM). Even though each stage in a CARE III model must use the same FEHM, because CARE III has only one FEHM, the parameter values used in the FEHM can be different for each stage. For each stage, a modeler inputs the different parameter values into the tool. Then CARE III automatically places the results of the appropriate FEHM solution on the appropriate transitions of the Markov Model created from the fault tree.

4.1.2.1 Workaround for Representing Warm Spares. A workaround for the inability to represent sparing strategies other than hot (i.e., active) spares is to bound the "true" reliability by using two different failure rates. For example, modeling the system using the active failure rates for all the components provides a lower bound on the system reliability, and modeling the system using the warm failure rates for all the components provides an upper bound on the system reliability. The reliability of the actual system design using the warm sparing strategy lies between these bounds. This workaround can be used for any fault tree-based tool which can represent only hot spares.

The workaround may not be successful if fault types other than permanents are modeled. This is because the bounds on the system reliability may be so great as to be useless to the modeler or evaluator (e.g., upper bound = 0.9987 and lower bound = 0.5761).

There is no workaround for the inability to represent dependencies in the CARE III model. Since these design features cannot be represented in the model, it must be realized that the tool's results do not accurately reflect the design's actual reliability. The amount of the inaccuracy cannot be determined, since the error is a function of how many dependencies are not modeled, the failure rates of the components in each dependency, the redundancy strategy used with the components, and the coverage values of the fault tolerance mechanisms.

Creating a fault tree for CARE III is simple and straightforward. In fact, for the example system, all modelers should generate the same fault tree as the one in Figure 10, even if warm spares are modeled. The greatest error source in model construction that we have observed is modelers' attempts to "cleverly" overcome the limitations imposed by the fault tree notation. In doing this, modelers usually introduce more error into the predictions than would occur if they did not attempt such workarounds.

4.1.2.2 Increasing The Number Of Fault Tree Stages. CARE III as currently designed can only solve models with fault trees of 70 or fewer stages. This number may be too small to solve a detailed design model of an OBS. However, this number can be changed by changing CARE III's software. Details of this change are available from CARE III's developers at NASA Langley Research Center (LaRC).

4.1.3 Determining Fault Arrival Rates

The fault arrival rates of each component must be determined for each identified fault class. Our example system has two fault classes: single permanent and single transient faults.

CARE III accepts as input fault arrival rates that follow either an exponential or Weibull distribution. For each component, a modeler "tells" the tool by an appropriate input parameter which of the two distributions to use for a component and then provides the appropriate values for the selected distribution as inputs.

The fault arrival rates used in a model depend upon the types of design components used. If modelers used the same components with the same fault tree and FEHMs, they should get the same results.

4.1.3.1 Determining Permanent Rates. A modeler can use MIL-HDBK 217 to determine the permanent fault arrival rate of an electronic component. MIL-HDBK 217 is the government handbook for use in calculating permanent fault arrival rates of electronic components in the government acquisition process. MIL-HDBK 217 assumes component fault rates follow an exponential distribution. This assumption appears valid, since experimental data indicate that permanent faults occur in electronic equipment at a near constant rate after the infant mortality period. Table 1 lists the permanent fault rates of the components in the example system as calculated using MIL-HDBK 217.

For other types of components that do not follow an exponential distribution, a modeler can use the Government Independent Data Exchange Program or the Nonelectronic Parts Reliability Data from Rome Laboratory to obtain the fault arrival rates of these components.

Table I. Failure Rates for Example System

| Subsystem | Failure Rate/$10^6$ hr |
|---|---|
| Clock | 0.1376 |
| Power Supply | 0.4529 |
| System I/O | 0.5428 |
| Nodal I/O | 0.7065 |
| Scalar Processor | 2.0095 |
| Array Processor | 3.0135 |
| Non-volatile Memory | 0.6018 |
| Random Access Memory | 1.5212 |

4.1.3.2 Determining Transient Rates. Transient faults can significantly impact OBS reliability, since these faults occur one to three ordes of magnitude more frequently than permanent faults. However, transient fault arrival rates are very difficult to accurately determine for use in modeling tools. It may be impossible to identify all the relevant design parameters affecting transient fault rates and to correctly determine their values. As a result, each design component should be modeled using a range of transient fault rate values (e.g., from ten to a thousand times its permanent fault rate). Modeling a design using a range of transient fault rates allows modelers to determine how sensitive individual components and design reliability are to transient faults.

An approach to narrowing the range of transient fault rates is the use of analytical models. These models use various design parameters (i.e., the OBS's orbit, shielding, and process technology) to predict the transient fault rate of design components. These models have been implemented as computer programs. Since these models may not include all the relevant design parameters that affect transients, the government and system developers should agree if these programs are to be used. This prevents misunderstanding about whether the model used in a program accurately includes design parameters that are relevant to the occurrence of transients in the actual design.

4.1.3.3 Justifications for Parameter Values. Because of the variance in determining fault arrival rates and in the rates themselves, it is necessary that more information be provided to independent design reviewers than just the final reliability numbers. The justification for selecting the parameter values (e.g., part quality, temperature) used in calculating the fault arrival rate should be provided along with the reliability predictions, so that the reliability predictions of each developer can be

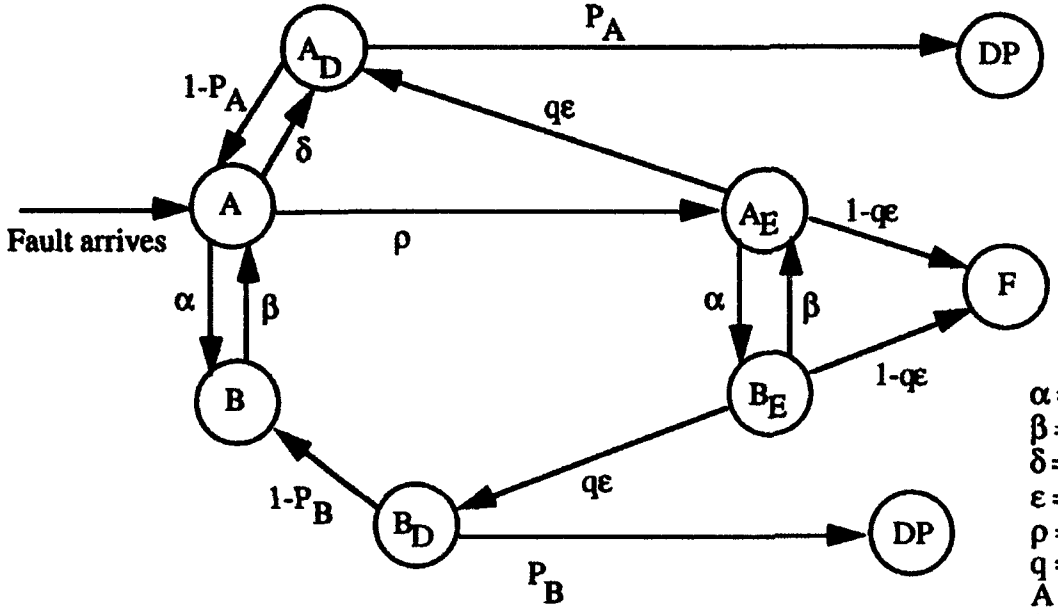
properly evaluated by the government. Also, early in the acquisition process, the government and the system developers should agree on the fault arrival rates to be used. Such an agreement avoids any potential misunderstandings and helps ensure a consensus on the values to be used.

### 4.1.4 Modeling Fault/Error Handling in CARE III

4.1.4.1 An Explanation of the CARE III FEHM. In this section, the CARE III FEHM, illustrated in Figure 11, is described. The CARE III FEHM represents the tool developer's abstraction of how designs generally handle faults and errors. The CARE III FEHM is a fixed tool feature (i.e., it cannot be changed). As a result, if this FEHM does not represent accurately how a design handles faults and errors, a modeler cannot change the FEHM to allow for more accurate modeling. A modeler must provide a value for each parameter in the FEHM.

The CARE III FEHM is the actual model used to determine the proba-bility of taking each exit in the general FEHM. Several of the states in the CARE III FEHM correspond to the exits in the general FEHM. The CARE III FEHM state labeled F corresponds to the S exit of the general FEHM. The CARE III FEHM state labeled DP corresponds to the C exit of the general FEHM. When transients are modeled, the CARE III FEHM state labeled B corresponds to the R exit of the general FEHM. No state in the CARE III FEHM directly corresponds to the N exit of the general FEHM. Although the FEHM does not provide an N exit, the tool takes the near-coincident fault rate and the FEHM parameters provided by the modeler and calculates automatically the probability of taking the N exit and includes this probability in the model solution.

When a fault occurs, the CARE III FEHM enters the active state (state A). In this state, three outcomes are possible. First, a fault may be detected by self-test, performed at rate $\delta$. If the fault is detected by self-test, state $A_D$ is entered. Second, a fault may generate errors at rate $\rho$. If an error is generated, state $A_E$ is entered. Third, a fault may become benign at rate $\alpha$. A benign fault is incapable of generating an error. If a fault becomes benign, state B is entered. If transient faults are being modeled, state B represents a system state where a transient fault no longer exists in a system. In state B, a benign fault other than transients can become "active" again (i.e., state A is entered) at rate $\beta$.




α = rate benign error state entered
β = rate active state entered
δ = rate of fault detection
ε = error detection rate
ρ = error generation rate
q = error detection coverage
A = active
P = benigh
D = detected
E = error
F = failure
DP = detected as permanent
$P_A$ = probability of faulty component being permanently removed from the system
$P_B$ = probability of faulty component being permanently removed from the system

Figure 11. CARE III FEHM

In state $A_E$, three outcomes are also possible. First, an error may be detected by the fault tolerance mechanisms. These mechanisms operate with an error detection rate $\epsilon$ and an error detection coverage of q. If the error is detected, state $A_D$ is entered. Second, if an error is not detected by the fault tolerance mechanisms, then the system fails (i.e., the F state is entered) at rate $(1-q)\epsilon$. Third, the error state may enter a benign fault state (i.e., state $B_E$) at rate $\alpha$. In this state, the fault is benign and cannot generate more errors, but one or more errors exist in the system.

Like the previously discussed state, the benign fault state has three outcomes. First, an error can be detected by the fault tolerance mechanisms operating with an error detection rate $\epsilon$ and an error detection coverage of q. If the error is detected, state $B_D$ is entered. Second, if the error is not detected by the fault tolerance mechanisms, then the system fails at rate $(1-q)\epsilon$. Third, in this state benign faults other than transients can become active again (i.e., state $A_E$ is entered) at rate $\beta$ and generate errors.

The error detected states (i.e., states $A_D$ or $B_D$) have two outcomes. First, the erroneous component is permanently removed from the system (i.e., state DP is entered) with probability $P_A$ or $P_B$. Second, with the complementary probabilities (i.e., $1-P_A$ or $1-P_B$), a faulty component is returned to the system where it may generate errors.

The handling of three different fault types can be modeled using the CARE III FEHM, depending upon the values of $\alpha$ and $\beta$. If $\alpha = 0$ and $\beta = 0$, then the handling of a permanent fault is modeled. If $\alpha > 0$ and $\beta = 0$, then the handling of a transient fault is modeled. If $\alpha > 0$ and $\beta > 0$, then the handling of an intermittent fault is modeled. For each different fault type, a separate FEHM is used. The tool solves each FEHM separately. Then, for each FEHM exit, the tool combines the separate exit probabilities into a single exit probability. This "combined" FEHM is then used in solving the models.

4.1.4.2 A Limitation of the CARE III FEHM. The CARE III FEHM cannot accurately model a system that always attempts transient recovery first on any fault. This type of system initially attempts a fixed number of transient recovery attempts before considering the fault permanent and then performing permanent fault recovery, since the system cannot discriminate initially between permanent or transient faults.

The inability to model this situation results in cases where transients that may be recovered from in an actual design are treated as permanents in CARE III. For example, let us assume a transient fault exists in a system and the CARE III FEHM is in state A. The transient

may generate an error and enter state $A_E$. The error is detected by the fault tolerance mechanisms and state $A_D$ is entered. Then, the component is considered permanently failed with probability $P_A$ and removed from the system. Thus, in the first attempt at recovery the fault was considered permanent, whereas in an actual design several other attempts at recovery would be attempted before permanent fault recovery is attempted. No workaround exists for this limitation.

4.1.4.3 FEHM Parameter Values for Use Early in the Design Cycle. In this section, determining the parameter values for use in the CARE III FEHM to get the reliability prediction process started is explained. Early in the design cycle, modelers perform analyses to determine the sensitivity of a design's reliability to coverage. To do this using CARE III, a modeler must provide values for the CARE III FEHM parameters. However, early in the design process, no design detail exists from which the FEHM parameters may be derived.

By using the following steps a modeler can determine what parameter values to use in the FEHM, so that the tool will derive the coverage value to be included in a model. Assuming that a fault has occurred, a modeler first sets $P_A = 1$. This assumption simplifies a formula used by a modeler to determine the parameter values. Second, the modeler chooses arbitrary values for $\delta$ and $\varepsilon$. Third, a modeler uses the values of $\delta$ and $\varepsilon$ in the formula $(\delta + q\varepsilon)/(\delta + \varepsilon) = c$ to determine an error detection coverage q for the desired derived coverage value c. The following paragraph illustrates the use of these steps.

Let us assume that a modeler desires to predict the reliability of an OBS design using an estimated overall coverage value of 0.99 (i.e, $c = 0.99$). First, the modeler sets $P_A = 1$. Next, the modeler chooses arbitrary values of $\delta$ and $\varepsilon$. In this example, the modeler lets $\delta = 100$ detections/hr and $\varepsilon = 200$ errors/hr. Now, the modeler solves the formula $(100 + q200) / (100 + 200) = 0.99$ for q; the modeler determines that $q = 0.985$. Thus, by using the values of $\delta = 100$, $\varepsilon = 200$ errors/hr, $P_A = 1$, and $q = 0.985$ for the FEHM parameter values, the modeler predicts the reliability of a design with a estimated overall coverage value of 0.99.

It is very important to note that these parameter values are only an artifice allowing the prediction process to begin. At this point, the values used for the FEHM parameters do not necessarily reflect the actual effectiveness of the fault-tolerance mechanisms that will be present in the actual design. Therefore, the models' results provide information about only "what-if" a design provided these values.

4.1.4.4 FEHM Parameter Values for Use Later in the Design Cycle. Later in the design cycle when design details regarding the handling of faults and their errors are available, simulations or emulations should be used to derive FEHM parameter values. The principal difficulty in doing this is that the people performing the simulations or emulations do not know exactly what to measure. For example, the rate at which a fault generates an error is a CARE III FEHM parameter. In OBSs, this rate depends upon what program is executing, what data the program is using, and where in the design hierarchy the error generation rate is measured. Therefore, several different error generation rates can exist for the same fault, and it is not clear which value should be used as a FEHM parameter.

The FEHM parameter values determined by simulation or emulation may not be accurate to several decimal places. For example, it may be impossible using current methods to determine an error detection coverage to more than three decimal places. Therefore, modelers should not use a tool's predictions to more than the number of decimal places in the least accurate parameter. Presently no solution exists to this problem. This is because no method exists for measuring accurately the system attributes to determine the parameters for the CARE III or any other FEHM. Currently, the best approach to this problem appears to be an agreement between the government and system developers as to how the parameter values are determined. Thus the tools' results using these parameter values are not absolute measures of design reliability but provide relative measures allowing the government to uniformly evaluate developer's efforts.

### 4.1.5 Model Review

After a model for any tool is constructed, the model should be reviewed by people knowledgeable about the design to determine if the model is accurately constructed [Triv 84]. A model review is the best way to minimize errors that may exist in a model because no automated techniques have been developed to determine if a model represents a design accurately. This is clearly an area for future work.

## 4.2 CONSTRUCTING A HARP MODEL

In this section, how to construct a HARP model of the example system is discussed. HARP also uses Markov models for the reliability prediction of OBSs. HARP, like CARE III, uses fault trees plus additional information to construct a Markov model representation of a design. However, there is a slight difference in the fault trees used by HARP, in that basic events are defined differently. In HARP, a basic event is the failure of an individual system component (i.e.,

the failures of processors, memories, and buses) instead of the failure of a group of identical system components (i.e., a stage). This slight difference does not impact the model construction of our example system using HARP by defining the conditions of failure down from the failure of a group of identical components to the failure of each individual component in that group. Therefore, modelers can follow the same three steps used to construct the CARE III model.

### 4.2.1 Constructing HARP Fault Trees

Because several versions of HARP may still be in use, it will be shown how to construct a fault trees for both the latest version (i.e., version 6) and the earlier versions (i.e., version 5 and earlier) as a group. After the presentation on fault trees, the remaining discussion on HARP is applicable to all the tool's versions.

4.2.1.1 HARP Version 5 and Earlier. To create a HARP fault tree for versions 5 and earlier, the conditions of failure for each design level are iteratively defined, from the highest level (i.e., system level) down to the lowest level (i.e., component failures). The previously mentioned logic gates are used to represent these conditions. The following paragraphs explain how to build the HARP version 5 fault tree for the example system.

The first step in building a HARP fault tree is defining the condition of system failure. Figure 12 shows the fault tree of just the system failure. Figure 12 shows that the event called "system failure" occurs if any subsystem fails. That is, the system fails if the clock OR the power supply OR the nodal I/O OR ··· the RAM fails.

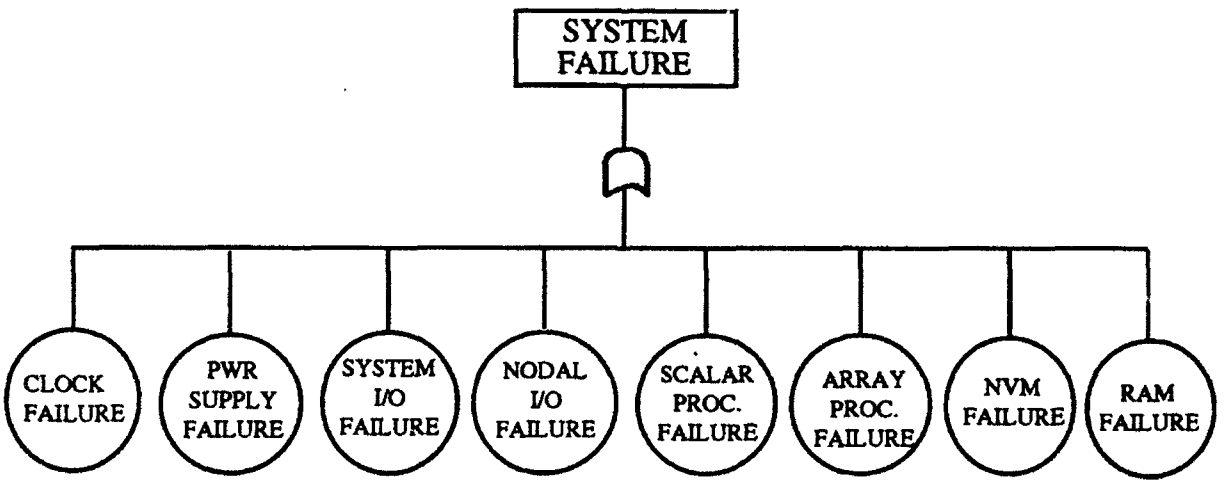



Figure 12. HARP Fault Tree of Example System

Next, the conditions of failure for each subsystem are defined. The conditions of failure for each design level continue to be defined until the failure of individual components are reached, rather than groups of identical components as in CARE III. The clock, power supply, system I/O, nodal I/O, and array processor each fails if both its active component AND spare fail. The processor subsystem fails if three out of the four processors fail. The NVM subsystem fails if two of the four non-volatile memories fail, and the RAM subsystem fails if three of the six random access memories fail. Figure 13 shows the completed fault tree of the example system. Since the design detail of the subsystem components is not yet fixed, the failure of subsystem components becomes our basic events.

The fault trees of these versions of HARP have the same limitations as the CARE III fault trees. That is, spares are considered active (i.e., hot) in the fault trees of version 5 and earlier, and the fault trees of these versions cannot represent functional and sequence dependencies. The workaround for modeling warm spares in CARE III is applicable to these versions of HARP.

HARP also allows modelers to overcome the limitations of the tool's fault trees by allowing Markov models to be input into the tool. The Markov models are more accurate because they allow design features to be represented that cannot be represented in a fault tree. The tool solves the Markov model and provides more accurate results. However, if the design is detailed enough to require a large Markov model (i.e., more than one hundred states), then the Markov approach is not recommended due to the potential for introducing mistakes because of the complexity of constructing the model.

A Markov model of an OBS design could be constructed very early in the design cycle, since the design is still very simple at this point. However, once detailed design begins, the required Markov model would easily require tens of thousands of states. Constructing large Markov models is very difficult and error prone. If a modeler did attempt to construct such a model, a tremendous amount of time would be required and the results would probably contain more error than the fault tree version. Therefore, even if a modeler used a Markov model to overcome the limitations of fault trees earlier in the design, a fault tree model must be utilized, with all of its limitations, to predict the reliability of the detailed design.

4.2.1.2 HARP Version 6. The latest version of HARP, version 6, provides new fault tree gate types that allow fault trees to represent hot, warm, and cold sparing, as well as functional and sequence dependencies. Since this is the first version of HARP to incorporate these gate types, the correct operation of these gate types has not been fully demonstrated. If these gates operate as

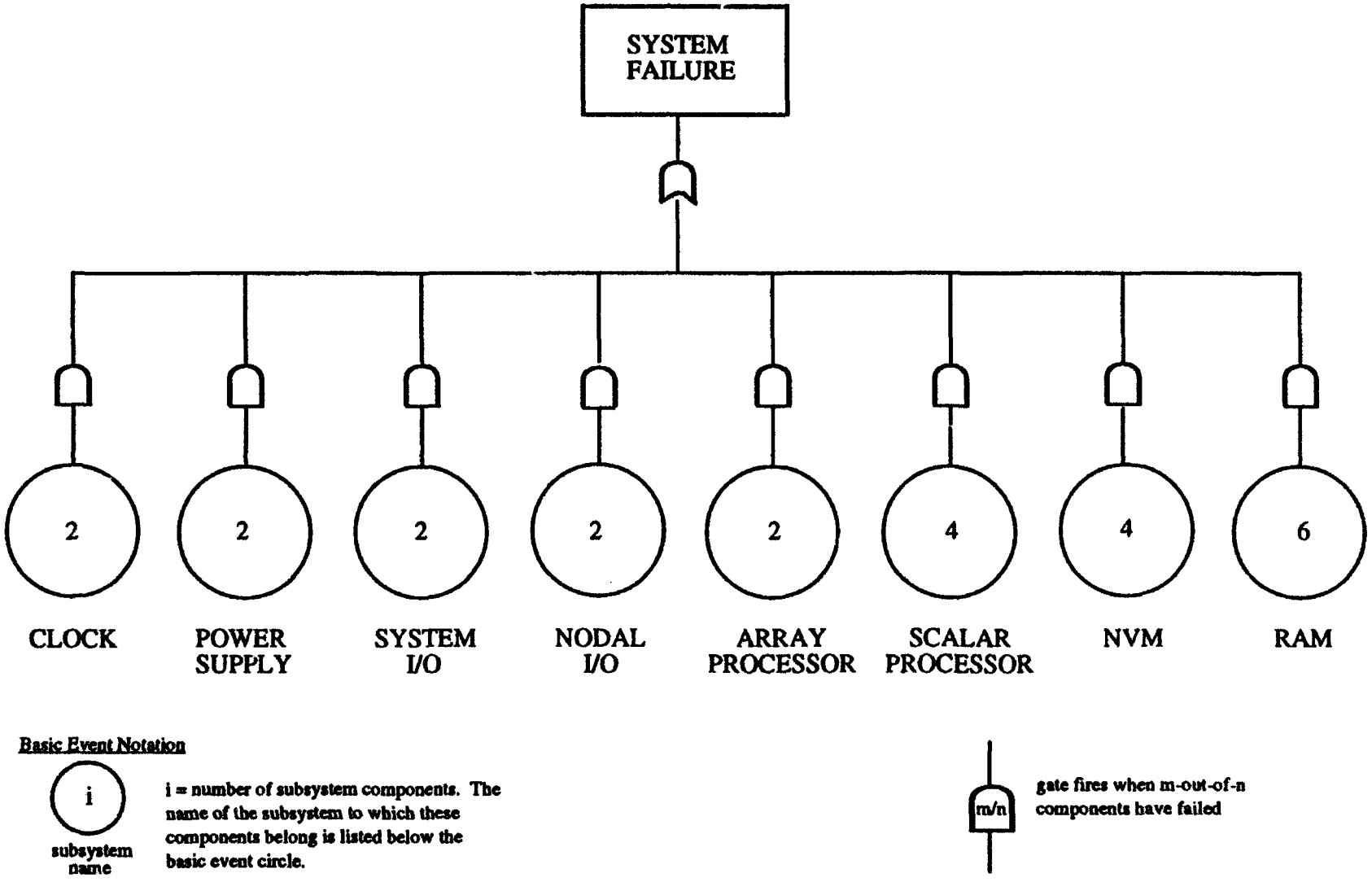



Figure 13. Complete Harp Fault Tree of Example System

claimed, modelers do not need to use Markov models or workarounds to represent these design features in the model. The tool generates a Markov model that represents the desired design feature when one of the new gate types is encountered during the construction of the Markov model from the fault tree.

Only two of the four new gate types, the dependency gate (FDEP) and the sequence enforcing gate (SEQ), will be discussed in connection with modeling the example system. The other two gate types, the cold spare gate and the priority AND gate, are clearly explained in [Duga 90] and the HARP User's Guide [HARP 89].

The functional dependency gate has one input, one or more dependent basic events, and one output. The input can be either a basic event (i.e., the failure of a component) or the output of any other fault tree gate, including the four new gate types. Once the input becomes active, the dependent basic events are forced to occur. However, the occurrence of any dependent basic event by itself does not force the input to become active. The output reflects the status of the input (i.e., active or not active) and can be used as an input to any other gate.

Figure 14 shows the use of a functional dependency gate for the example given in Section 1.4. When C occurs, the FDEP gate is activated. The gate now forces the dependent basic events (A and B) to occur.

The sequence enforcing gate may have any number of inputs. Each input is a basic event or an output of any other gate. In SEQ gates, the input events are forced to occur in the left-to-right order (i.e., the leftmost event must occur before the event on its immediate right which must occur before the event on its immediate right, etc.) The output activates only when all the inputs have activated. Also, the SEQ gate is unlike other fault tree gates in that when the left-most event occurs, the gate activates an "invisible" control line that enables the event on the immediate right to occur.

Figure 15 illustrates the use of a sequence enforcing gate for the first type of dependency described in Section 1.4. Once both computers in set A have failed, an input to the sequence gate is triggered. The triggering of the left-most input now activates the SEQ gate's "invisible control line." This control line enables the basic event on the immediate right to occur. Thus, the computers in set B can now fail. Once both computers in set B fail, the sequence gate "fires" to signal the failure of the subsystem.

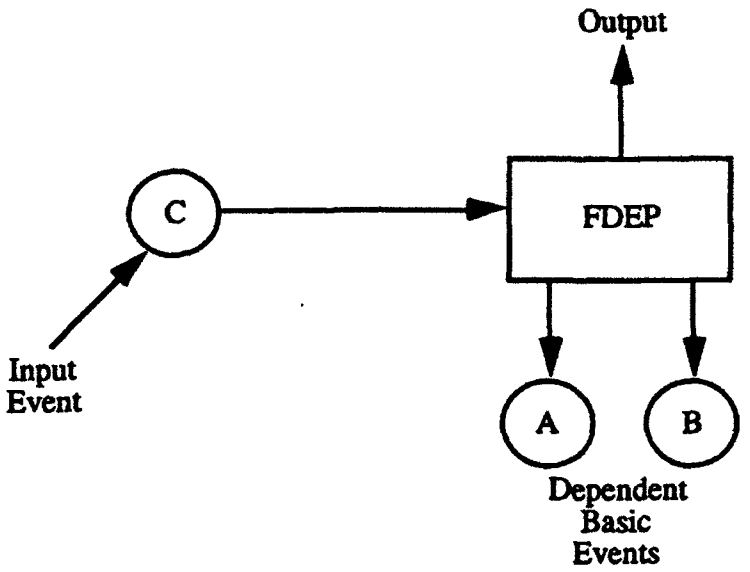



Figure 14. A Functional Dependency Gate

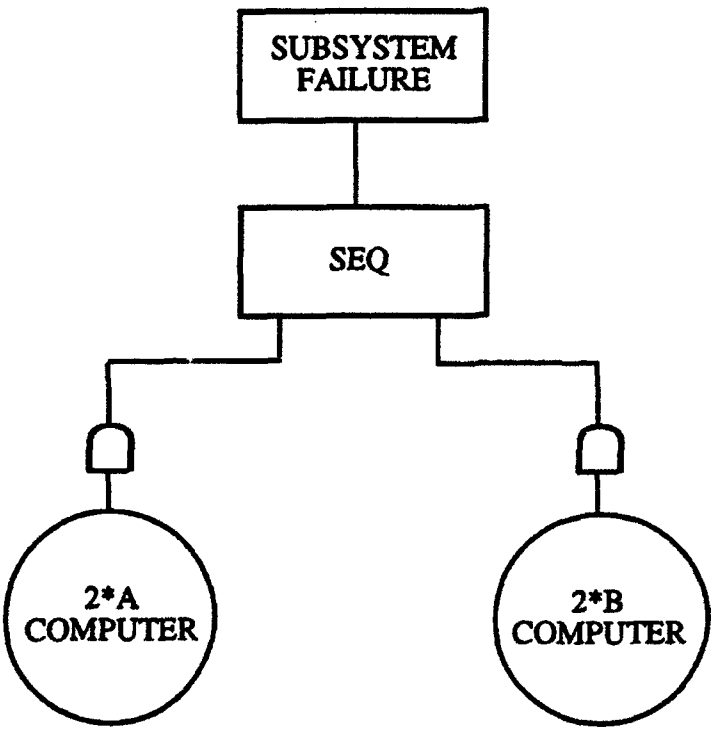



Figure 15. A Sequence Enforcing Gate

Constructing a HARP version 6 fault tree for our example system is the same as constructing a CARE III fault tree. That is, the conditions of failure are iteratively defined for each design level. Therefore, the fault tree of system failure is the same as the one in Figure 12.

To model more accurately the subsystems of the example system, the FDEP and SEQ gates can be used to represent the existence of warm spares in the subsystems. To do this, the ability to represent the changes in the number of spare components as a result of active or spare component failures is needed. The following paragraphs describe how this is done using a fault tree that represents a subsystem with three active and two spare components.

Figure 16 shows the HARP version 6 fault tree of a subsystem with three active and two warm spare components. The subsystem fails if three active components are not properly functioning. Initially, the only basic events that can occur are basic events 1 and 2. Once an active or spare component fails, OR gate A is triggered. This gate activates FDEP gate B and enables SEQ gates C and D. Activating FDEP gate B forces all the components in basic events 1 and 2 to fail. However, the system has not failed. This gate only serves as an artifice to remove the initial system configuration from the fault tree. Because their left-most inputs have been activated, SEQ gates C and D's invisible control lines now enable basic events 3 and 4. Two sequence gates are used here because we want to enable both basic events at the same time and not in a sequential order. Basic events 3 and 4 are now the only events that can occur, and they represent a subsystem configuration where only one spare is available.

The failure of one of the three active components or the remaining spare component (basic events 3 and 4) activates OR gate E. This OR gate in turn activates FDEP gate F and SEQ gate G. Activating FDEP gate F forces basic events 3 and 4 to occur, thereby removing that subsystem configuration from the fault tree. Activating the left-most input of SEQ gate G results in its invisible control line enabling basic event 5, which is now the only basic event that can occur. This event represents the operation of only the three active components.

Basic event 5 occurs when any of the three active components fails. Upon the occurrence of this event, the OR gate (gate H) activates, and the subsystem fails.

This approach is used to represent the warm sparing in each subsystem of the example system. Figure 17 illustrates the resulting fault tree.

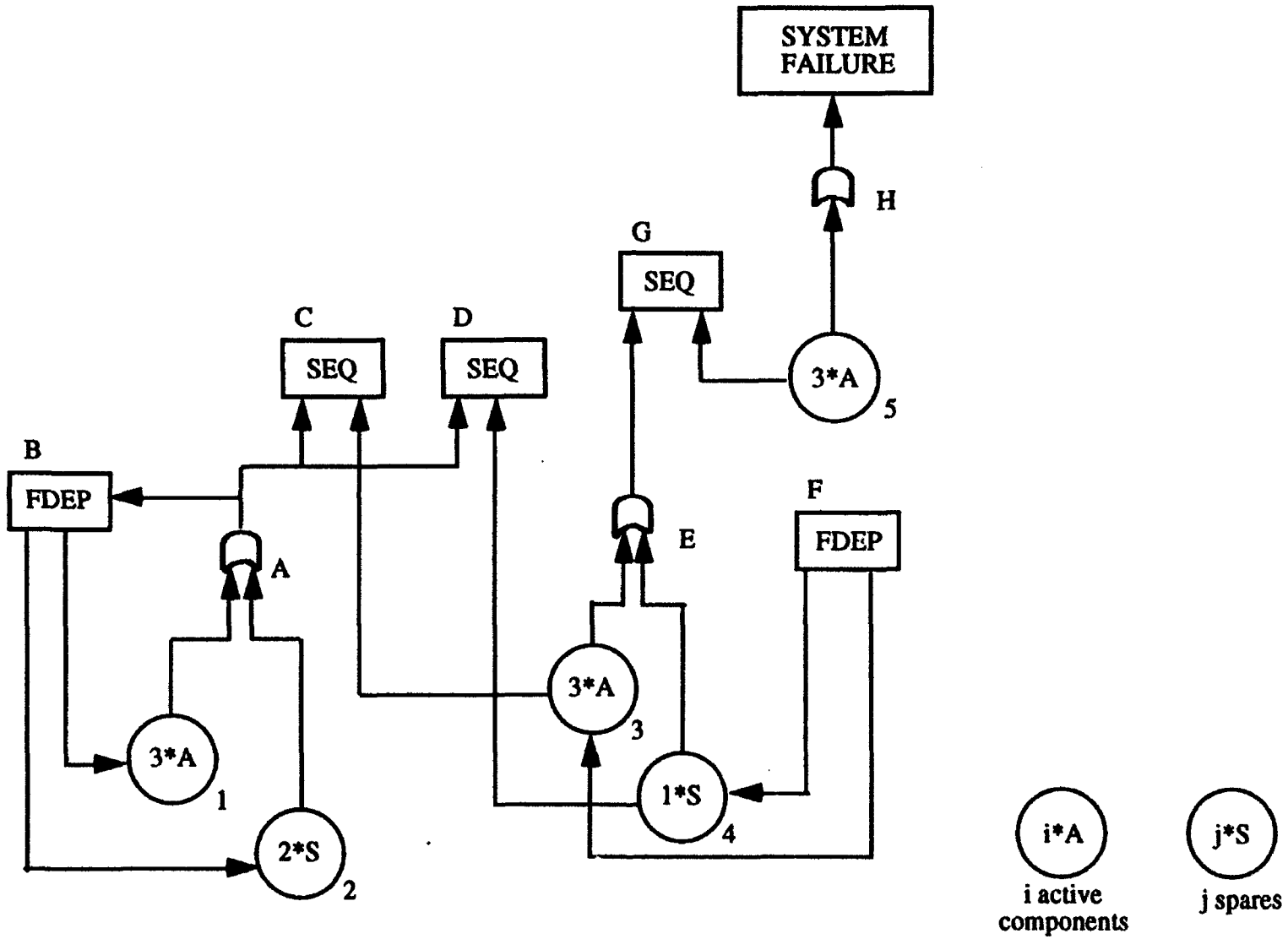



Figure 16. A Harp Version 6 Fault Tree of 3 Active and 2 Warm Spare Components

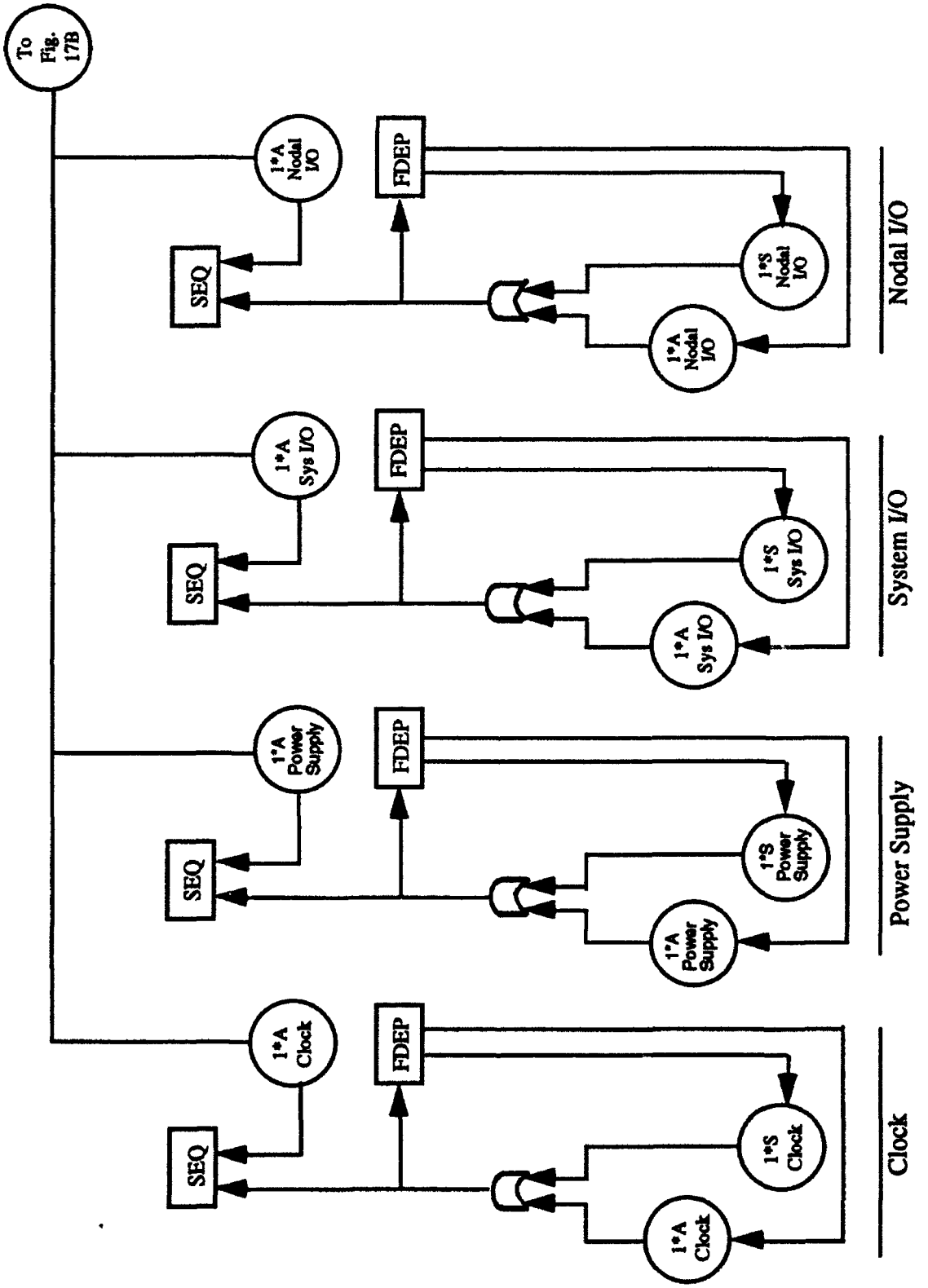



Figure 17A. A Harp Version 6 Fault Tree of the Example System

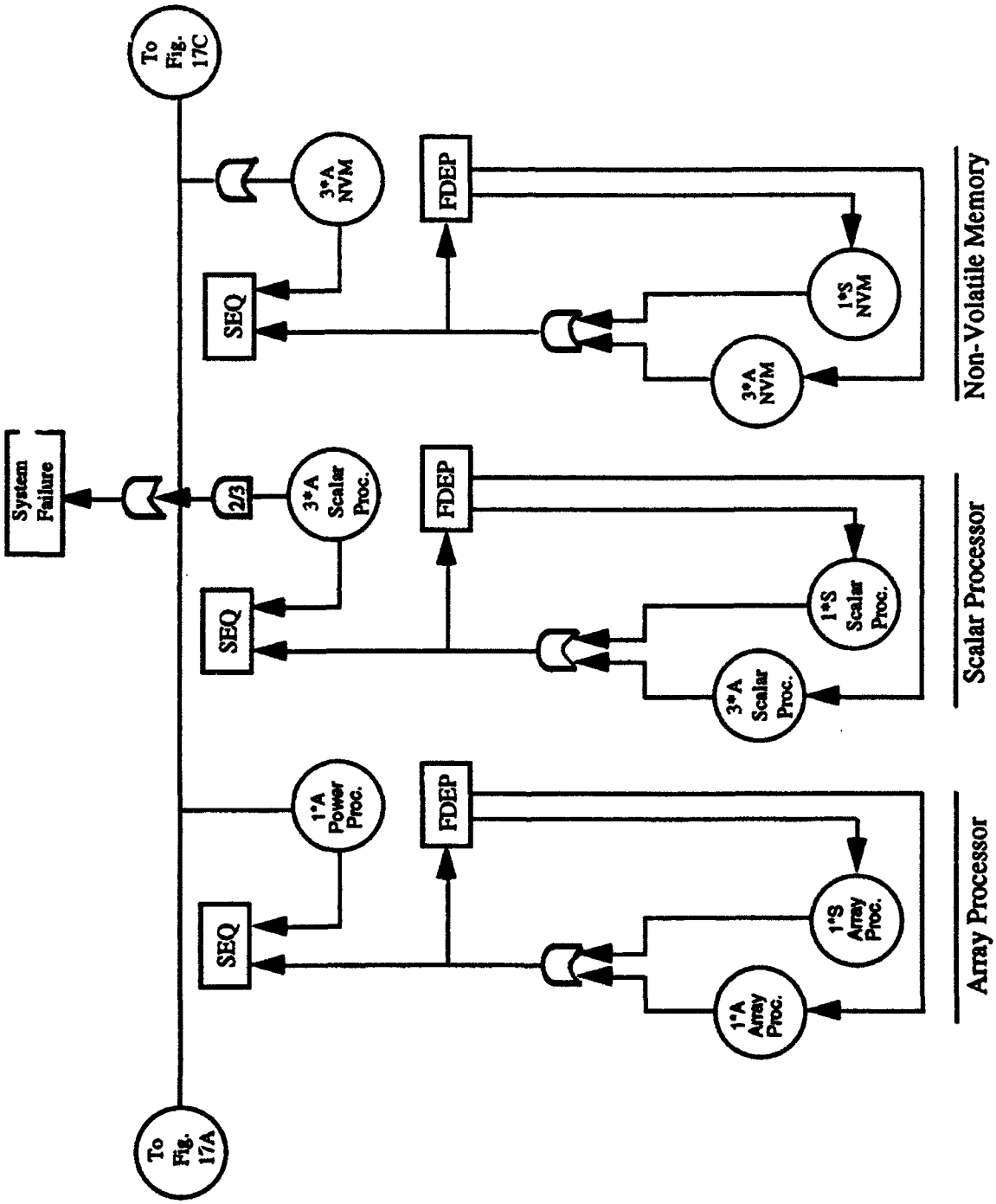



Figure 17B. A Harp Version 6 Fault Tree of the Example System

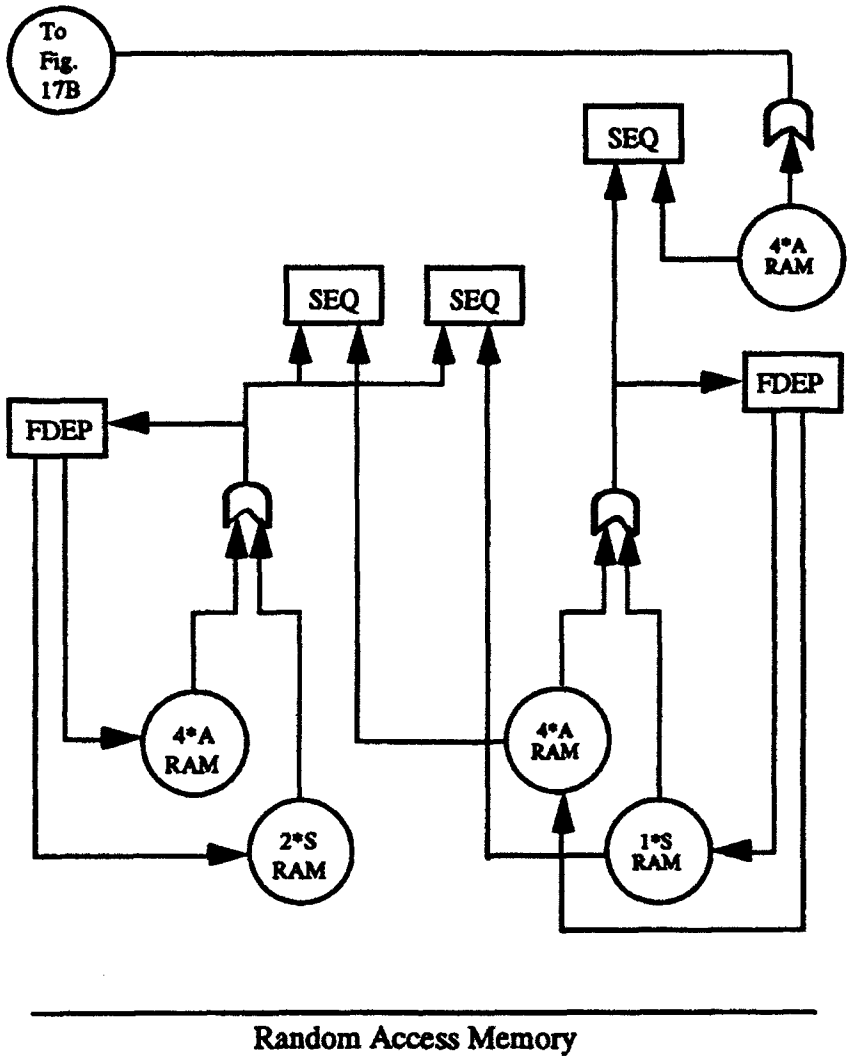



Figure 17C. A Harp Version 6 Fault Tree of the Example System

Although the new fault tree is more complex, the model is more accurate. In the new fault tree, the warm spares are treated as separate com-ponents. As a result, the spares can have a lower permanent fault arrival rate, and they are unaffected by transients. This modeling of spares reflects more accurately the treatment of spares in the actual system. Therefore, a more accurate prediction of the system's reliability is provided. Also, fewer model runs are required, since the model does not need to be solved twice as would be necessary if using a warm spare workaround.

### 4.2.2 Determining Fault Arrival Rates

The fault arrival rates for a HARP model are determined in the same way as those for CARE III. HARP, like CARE III, accepts as input fault arrival rates that follow an exponential or a Weibull distribution.

### 4.2.3 HARP's Fault/Error Handling Models

In this section, advice is given to modelers about using HARP's FEHMs throughout the design cycle. HARP provides modelers with eight FEHMs, including the CARE III FEHM, the ARIES FEHM [Ng 76], and the Extended Stochastic Petri Net (ESPN) FEHM [Dug 84], to model the details of a design's fault and error handling behavior. For an in-depth discussion of each FEHM, the reader is referred to the HARP User's Guide and [Duga 86].

Throughout the design cycle, the modeling of a design's fault and error handling behavior changes. Early in the design cycle, arbitrary coverage values (e.g., 0.95) should be used in place of FEHMs. This is because no design details exist regarding the handling of faults and their resulting errors. The use of arbitrary coverage values frees a modeler from the need to consider the specific details of modeling a design's fault and error handling behavior. Also, the use of arbitrary coverage values allows the modeler to perform parametric studies of the impact of various coverage values on design reliability.

As the details of how a design handles a fault and its errors evolve, so does the modeling detail of the FEHMs used in a model. For example, when the transient fault handling mechanisms have been identified, the arbitrary coverage values can now be replaced with a more detailed FEHM (e.g., the ARIES FEHM). FEHMs more detailed than ARIES should not be used in the early design phases. This is because the design information and parameter values needed by the more detailed FEHMs (e.g., the ESPN FEHM) may not yet be available.

Once all the design's fault and error handling mechanisms are identi-fied, the most detailed FEHMs should be used (e.g., the ESPN FEHM). Certain of the FEHMs parameters may not yet be available from the design. Parametric analyses on those values will provide feedback to evaluators as to what these values should be for the design to meet its goals.

Any time during the use of a particular FEHM, if updated parameter values become available, these values must replace the values currently being used in the FEHM. Finally, actual measurements of the design can be used as FEHM parameters to provide the final predictions regarding design reliability.

## 4.3 CONSTRUCTING A CRAFTS MODEL

CRAFTS is a reliability prediction tool based on ARIES [Maka 82]. CRAFTS predicts the reliability of OBSs by using a mixture of modeling techniques (i.e., a combination of Markov models and RBDs). CRAFTS uses Markov models to model subsystems at the lowest design level, and the results of these subsystem models are used as inputs to a system-level RBD, which the CRAFTS tool uses to predict the OBS's overall reliability.

A modeler inputs an OBS model into the tool in two parts: one part for the Markov models and the other part for the RBDs.

In the first part, a modeler provides parameters describing each subsystem at the lowest design level. The tools uses these parameters to build the Markov model for each subsystem. In the second part, the modeler provides different parameters allowing the tool to build the system-level RBD, which the CRAFTS tool uses to combine the results of the Markov models to predict a design's overall reliability. To construct a model of the example system as input into CRAFTS, a modeler must:

a. Determine the failure rates of the lowest level components.
b. Determine the parameter values to be used in the fault/error handling model of each lower level component.
c. Determine the parameter values required by the tool to construct a Markov model of each group of identical components at the lowest design level.
d. Build a reliability block diagram of the system design.

CRAFTS' use of the two nomenclatures prevents functional and sequence dependencies from being represented in the resulting model. In the first part, no parameter(s) are provided allowing a modeler to descibe the sytem dependencies, thus the tool cannot reflect the

dependencies in the resulting Markov model. In the second part, reliability block diagrams simply cannot represent these design features.

However, CRAFTS provides a potential workaround to this problem. CRAFTS, like HARP, has another nomenclature that allows modelers to directly input their own Markov models into the tool. Thus, a modeler can input a Markov model of a system or subsystem that accurately reflects these design features. The tool then solves the Markov model and provides the user with more accurate results. This workaround is not recommended if the system or subsystem model contains over a hundred states because of problems in constructing the model accurately.

### 4.3.1 Determining Fault Arrival Rates

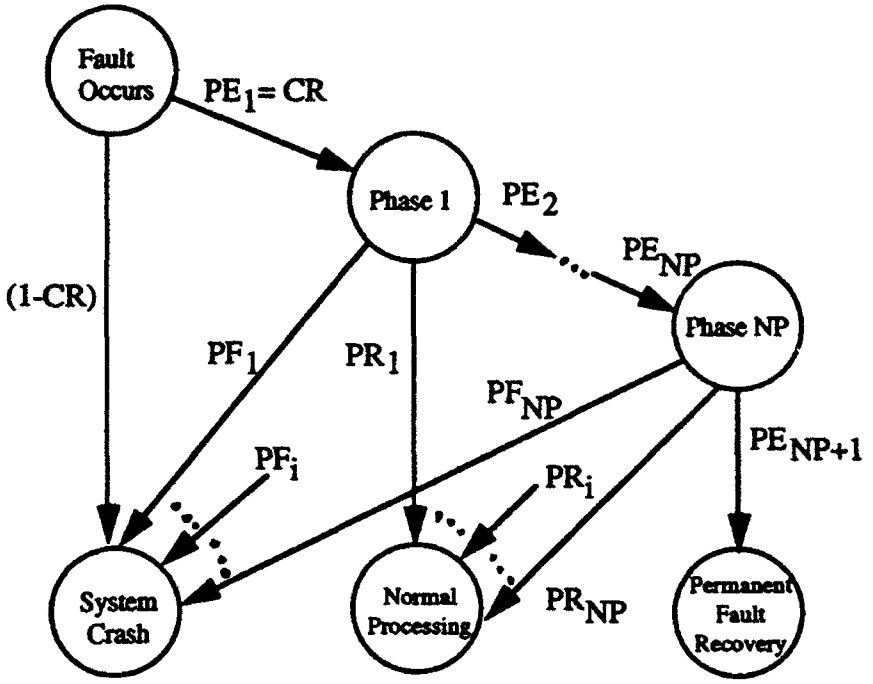
The fault arrival rates for a CRAFTS model are determined in the same manner as CARE III and HARP. CRAFTS only accepts fault arrival rates that follow an exponential distribution. As a result, the tool cannot be used to model components whose fault arrivals do not follow an exponential distribution. No workarounds exist to overcome this limitation.

### 4.3.2 Modeling Fault/Error Handling in CRAFTS

To model the fault and error handling behavior of the lowest level design components, CRAFTS uses the FEHM from ARIES. This FEHM, like the one in CARE III, is a fixed tool feature that cannot be changed. Therefore, if this FEHM does not accurately represent how a design handles faults and errors, a modeler cannot change the FEHM to allow for more accurate modeling.

The ARIES FEHM, illustrated in Figure 18, models designs that attempt transient recovery several times before beginning permanent fault recovery. The number of phases (NP) (i.e., attempts at transient recovery) is a FEHM parameter that is input by a modeler. Thus, the FEHM represents the exact number of transient recovery attempts present in a design.

Several of the states in the ARIES FEHM correspond to the exits in the general FEHM. In ARIES FEHM, the state labeled "system crash" and the permanent fault recovery that is unsuccessful corresponds to the S exit of the general FEHM. Successful permanent fault recovery in the ARIES FEHM corresponds to the C exit of the general FEHM. The state labeled "normal




CR = probability that a fault is not catastrophic
NP = number of phases in transient fault recovery
$PR_i$ = probability that the system returns to normal processing as a result of recovery phase i
$PF_i$ = probability of system crashing during phase i
$PE_i$ = probability of having to perform the next recovery phase

Figure 18. The Aries FEHM

processing" corresponds to the R exit of the general FEHM. For the same reasons as the CARE III FEHM, no state in the ARIES FEHM corresponds directly to the N exit of the general FEHM (see Section 4.1.4.1).

In addition to the number of phases used in the transient recovery, this FEHM requires a modeler to input the recoverability of a fault, the duration and effectiveness of each transient recovery phase, and the duration of a transient fault. The recoverability (CR) of a fault is the probability that the fault is not catastrophic. A catastrophic fault causes such massive damage that the system immediately fails. The duration of each transient recovery phase is constant, and the duration may be different for each different phase. The effectiveness of any phase i, $PR_i$, is the probability that the system returns to normal processing. The duration of a transient fault is the average length of time that a transient exists in a system.

The FEHM uses the effectiveness of each phase, the duration of each phase, and the duration of a transient to determine two other probabilities needed for each phase in the FEHM. The first probability, $PF_i$, is the likelihood of the system crashing (i.e., failing) during phase i. The second probability, $PE_i$, is the likelihood of having to perform the next transient (or permanent) recovery phase after phase i.

When a fault occurs, the ARIES FEHM enters the fault occurs state. In this state, a transition occurs to the first transient recovery phase or the system crash state. A transition to the first recovery phase occurs with the probability of the fault not being catastrophic (i.e., with probability CR). A transition to the system crash state occurs with the complementary probability (i.e., 1 - CR). Then for each recovery phase i, starting with the first phase and ending with the last phase, three possible transitions can occur. A transition to recovery phase i + 1 occurs with probability $PE_{i+1}$. A transition to normal processing occurs with probability $PR_i$, and a transition to the system crash state occurs with probability $PF_i$. For example, transition from the first phase to the second phase occurs with probability $PE_2$. Transition from the first phase to the normal processing state occurs with probability $PR_1$, and transition from the first phase to the system crash state occurs with probability $PF_1$.

If, after completing the last recovery phase (Phase NP), the system has not failed or resumed normal processing, the fault is considered permanent and permanent fault recovery is initiated. Although it is not shown in the illustration of the FEHM, if the permanent fault recovery is unsuccessful, the system fails.

The ARIES FEHM does not initially distinguish between transient and permanent faults. The FEHM always attempts transient recovery first on any fault. This is because a "real" system cannot usually distinguish between permanent and transient faults. As a result, the system initially "assumes" that a fault is transient, since transient faults occur much more frequently than permanent faults. If after all the transient recovery attempts the fault recovery is unsuccessful, the system considers the fault permanent and permanent fault recovery is performed.

### 4.3.3 Determining Parameter Values to Represent Subsystems as Markov Models

A modeler uses the first nomenclature to input the parameters that describe the structural properties (e.g., the number of active and spare com-ponents in a subsystem) and dynamic properties (e.g., fault arrival rates and fault and error handling behavior) of a group of identical components at the lowest design level. CRAFTS then uses the values of these parameters to construct a Markov model. An advantage of CRAFTS is that the tool can represent all the sparing strategies in a Markov model. This section discusses how structural parameters are determined. The parameter values used for the dynamic behavior are determined in the fault arrival rate and fault/error handling sections. For the example system, subsystems are treated as groups of identical components at the lowest level of the design, since the design details are incomplete.

Four parameters represent the structural properties of each subsystem in the example system. These parameters are N, S, D, and $\overline{N}$. N is the number of active components in each subsystem. S is the number of spares in each sub-system. D is the number of degradations (i.e., failures) allowed in the number of active components before the subsystem fails. The number of active components in a subsystem is not allowed to degrade until after the spares in that subsystem are exhausted. $\overline{N}$ is a vector that represents the number of active components in each degraded state. For example, if D =2 (i.e., there are two degraded states), then $\overline{N}$ is a two-dimensional vector.

A hypothetical subsystem that consists of four active processors and no spares is used to illustrate how these parameter values are determined. For the subsystem to correctly function, two of the four processors must work. Since there are no spares, S = 0. Since there are four active processors, N = 4.

Upon the first processor failure, the subsystem can degrade and con-tinue to correctly operate with three computers. Once a second processor fails, the subsystem can degrade again and continue to correctly function with two processors. Since the subsystem can tolerate two

degradations (i.e., processor failures) and still operate, D = 2. After the first degradation, three processors continue to work; therefore $\overline{N}[1] = 3$. After a second degradation, two computers still function correctly; therefore, $\overline{N}[2] = 2$. In summary, the structural parameters for this hypothetical subsystem are N = 4, S = 0, D = 2, and $\overline{N} = 3, 2$.

The number of active components in the $\overline{N}$ vector is not necessarily decremented by one from the number of active components in the previous degraded state or from the initial number of active components. For example, if a subsystem has three active components, the subsystem may degrade to a simplex upon the failure of one active component. Thus the value of $\overline{N}[1]$ would be one and not two.

Table 2 shows the structural parameter values for each subsystem. The clock, power supply, system I/O, nodal I/O, and array processor subsystems each have an active and a spare component. Therefore, N = 1 and S = 1 for these sub-systems. D and $\overline{N}$ equal zero because they cannot degrade. That is, once a spare is no longer available and the active component fails, the system fails. For the scalar processor subsystem, N =3, S = 1, D =1, and $\overline{N}[1] = 2$. This is because the processor initially has three active processors and one spare. Once the spare is no longer available and one of the three active processors fails, the processor subsystem can degrade to two active processors. For the NVM, N =3, S =1, D =0, $\overline{N} = 0$. The NVM has three active memory modules and a spare. If the spare is unavailable and an active memory module fails, the NVM subsystem fails. The RAM subsystem has four active and two spare memory modules. Once the spares are unavailable and an active module fails, the RAM subsystem fails. Therefore, for the RAM subsystem, N = 4, S = 2, D = 0, and $\overline{N} = 0$.

Table 2. CRAFTS Parameters for Example System

| Subsystem | N | S | D | $\overline{N}[D]$ |
|---|---|---|---|---|
| Clock | 1 | 1 | 0 | 0 |
| Pwr Supply | 1 | 1 | 0 | 0 |
| System I/O | 1 | 1 | 0 | 0 |
| Nodal I/O | 1 | 1 | 0 | 0 |
| S. Processor | 3 | 1 | 1 | 2 |
| A. Processor | 1 | 1 | 0 | 0 |
| NVM | 3 | 1 | 0 | 0 |
| RAM | 4 | 2 | 0 | 0 |

### 4.3.4 Constructing the Reliability Block Diagram

Building a reliability block diagram (RBD) for CRAFTS begins by constructing a RBD of the top-level in a system. Next, take each block in the top-level RBD and create an RBD of the components comprising that block. Iteratively repeat this procedure for each block in each RBD until the blocks in each reliability block diagram represent a group of components at the lowest level of the design. At that point, the second nomenclature is used to input the RBDs into CRAFTS.

Figure 19 shows the top-level reliability block diagram of our example system. The system fails if any subsystem fails. This is because a path from source to sink cannot be traced if a subsystem is removed because of failure. Since our subsystems are the groups of components at the lowest design level, no RBDs of the subsystems need to be constructed, and the reliability block diagram of the example system is complete.

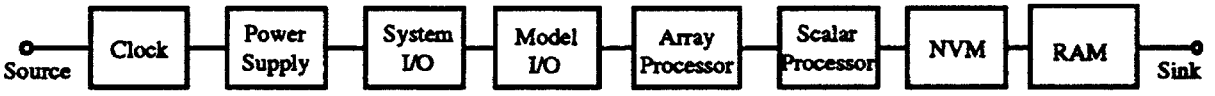



Figure 19. Top Level Reliability Block Diagram

## 4.4 Section Summary

In Section 4, how to correctly model the example system using CARE III, HARP, and CRAFTS was shown. For each tool, the limitations preventing the accurate modeling of the example system were also shown. For the tool limitations for which no workaround exists, reduced accuracy is the result. The next section discusses how to select a reliability prediction tool so that tool limitations are minimized.

# 5 SELECTING TOOLS

A modeler's first problem in reliability prediction is selecting tools that allow construction of the most accurate models of a design. Tool selection is important because it allows modelers to select tools with the fewest limitations which a` priori limit the accuracy of model construction.

To select tools, each tool's ability to accurately represent a design's structure, fault arrival process, fault/error handling, and correct coverage treatment should be evaluated. A design's structure is a list of the design's components, their interconnection, and conditions under which a design fails. Design structure is also concerned with what redundancy strategies (e.g., static, dynamic, or hybrid) and sparing schemes (e.g., hot, warm, or cold) are used in a design's components. The fault arrival process deals with both the fault types in a design and the distributions of those fault types. Fault/error handling is the way a design handles a fault and its resulting errors. A tool allows correct treatment of coverage when coverage values can be included at every design level.

In the following paragraphs, the best tools to model the example system are determined by comparing the tools in each of these areas. Any reference to HARP refers to version 6, the most recent version.

## 5.1 DESIGN STRUCTURE

Accurate modeling requires that the selected tools provide the capability to represent accurately the different redundancy strategies and sparing schemes present in the example. All three tools have the ability to represent accurately the different redundancy strategies used in the example system and hot spares. However, warm or cold spares may be used in the example system. HARP and CRAFTS can accurately model these types of spares, but CARE III cannot.

No capability to model accurately functional or sequence dependencies is needed, since these dependencies do not exist in the example system. However, if functional or sequence dependencies existed in the example system, only HARP provides the ability to represent them in a detailed model. The workaround in CRAFTS to represent these dependencies is not useful for detailed models because of problems in accurate construction of the required model.

## 5.2 FAULT ARRIVAL PROCESS

To model our example system accurately, the selected tools must provide the ability to model single permanent and single transient faults, and near-coincident faults. All of the available tools have this ability. The inability of CRAFTS to model components whose fault arrival rates do not follow an exponential distribution is not a limitation, since components of that type do not exist in the example system.

## 5.3 FAULT/ERROR HANDLING

To predict the reliability of our example system accurately, the selected tools must provide the ability to accurately represent the behavior of a design in handling transient and permanent faults and their resulting errors. The accurate modeling of transient errors is important, since it can have the greatest impact on design reliability.

For purposes of illustrating the comparison of tool FEHMs, it is assumed that the example system has a transient recovery process that consists of a fixed number of recovery phases. Each phase has both a fixed time in which to recover from the fault and an effectiveness in recovering from the fault. The CARE III FEHM cannot model accurately this type of transient fault handling. The ARIES FEHM that is present in both CRAFTS and HARP accurately models this type of transient recovery. Additionally, HARP's ESPN FEHM can be used to model our example system's transient fault handling.

We also assume that our example system will contain mechanisms allowing recovery from near-coincident faults. However, all the tools consider the occurrence of a near-coincident fault as a system failure. As a result, the example system's handling of a near-coincident fault cannot be accurately modeled. Because a near-coincident fault is treated as a system failure, the tools' results are conservative.

## 5.4 CORRECT COVERAGE TREATMENT

Accurate modeling of our example system requires that the selected tools allow coverage values to be included in the model at any design level, since the example system will have hierarchical treatment of and recovery from errors. However, all the available tools allow coverage to be included only at the lowest design level. As a result, the selected tools will provide optimistic predictions of our example system's reliability.

## 5.5 SELECTED TOOLS

HARP and CRAFTS are the tools recommended for predicting the reliability of the example system. CARE III cannot represent warm or cold spares, or the transient recovery process used in the example system. Both HARP and CRAFTS can accurately model the use of warm or cold spares in the model of the example system. Also HARP and CRAFTS can model the details of the transient fault handling that exists in the example system.

The use of both HARP and CRAFTS is recommended because the use of two tools instead of only one provides confidence that correct results are being provided by comparing results. If an error occurred in constructing either tool's model, the error should be detected, since the two tools use different model construction techniques (i.e., fault trees vs. reliability block diagrams). Also, if an error occurs in either tool's solution process, that error will be detected, since the two tools use different model solution techniques (i.e, ordinary differential equations vs. eigenvalues).

# 6 IMPROVING RELIABILITY PREDICTION IN THE DESIGN CYCLE

In the following sections, how errors can be further reduced in the reliability prediction process are identified . In the first five sections, improvements to the government specification of the reliability prediction process are discussed. In the last four sections, improvements that allow reliability prediction tools to model an OBS more accurately are identified. These improvements will enable the tools to provide more accurate feedback to evaluators about design upgrades.

## 6.1 MODIFY MIL-STD 756B

To predict more accurately the reliability of OBSs, MIL-STD 756B [MIL 81] should be modified to include reliability prediction techniques that allow more accurate models of OBSs to be constructed. MIL-STD 756B is the principal contractual document used to describe the techniques system developers must use in performing reliability prediction. However, the reliability prediction techniques described in the document were developed before OBS designs emerged. As a result, the techniques in MIL-STD 756B do not allow the construction of models that are good approximations to OBSs (e.g., functional dependencies cannot be modeled). As an initial guideline, MIL-STD 756B should be modified to include reliability prediction techniques with the capability to model the following important OBS design features and characteristics:

a. Coverages and fault/error handling
b. Permanent, transient, intermittent, and near-coincident faults
c. Degraded system operation
d. Masking and dynamic redundancy
e. Hot, warm, and cold spares
f. Exponential and Weibull fault rate distributions
g. Functional dependencies
h. Sequence dependencies

## 6.2 VERIFY RESULTS

Requiring the use of a second reliability prediction tool or method will allow reliability prediction results to be verified. The verification of modeling results is fundamental to any engineering study. Unfortunately, system developers in general use only one tool or method for reliability prediction. Developers should use a second tool or method based on different mathematics, model construction techniques, and model solution approaches to provide result verification. In the author's opinion, the unreliability generated by the two tools or methods should not differ by more than 5%. The second tool or method can be another reliability prediction

tool (such as the developer's own tool or a publicly available tool) or a viable reliability prediction method (such as Monte Carlo simulation).

## 6.3 OBTAIN MODELS

In addition to the reliability prediction results, system developers should provide government evaluators with descriptions of the actual reliability prediction models, so evaluators have a proper context in which to assess the results. Independent system evaluators need the models to ensure that they are correctly constructed, that reasonable assumptions have been made in the models' construction, and that model parameters are correctly determined.

## 6.4 CALIBRATE RELIABILITY PREDICTION PROCESS

The government should require system developers to predict the reliability of an example OBS design, so developers' capability can be assessed. If any error sources are identified in the example reliability prediction, then the system developer can take steps to remove or minimize these error sources before predictions of the actual system begin.

## 6.5 EVALUATE PREDICTION EFFORTS THROUGH QUESTIONING

Included below is a baseline set of questions that can serve as a qualitative tool to help improve the reliability prediction process through a better understanding of the components of the process. The questions address the areas of model construction, modeling analyses, and modeling verification. It is recommended that these questions be used by both the contractors and the government evaluators.

### 6.5.1 Model Construction

The model construction questions below address the issues of how accurately the structure, error handling processes, and identified fault set of the design have been mapped onto the design model. These questions also ask for the assumptions in the model to be identified and justified. In addition, the questions ask for justification of the values being used for the fault arrival rates and error handling behavior parameters:

a. Does the mapping of the system design to the model illustrate:
   1. all system components
   2. how they are interconnected

3. the conditions under which the system fails

b. What dependencies, if any, exist among the system components and how are they reflected in the model?

c. For each design component, what are the fault types modeled and what failure rate distribution was selected to model those faults? Discuss briefly why the given distribution was selected

d. How were the permanent failure rates determined? If MIL-HBK 217 was used in calculating the permanent failure rates, what assumptions were used in the calculation (i.e., part quality, temperature)? What is the justification for those assumptions?

e. How are the values for transient and intermittent faults determined?

f. How are near-coincident faults being modeled? What system components can be affected by near-coincident faults and why? If near-coincident faults will not be modeled, justify that decision.

g. How was the FEHM for each component selected? Explain how the fault/error handling processes (error detection, fault diagnosis and isolation, and recovery methods) of each system component were mapped onto the chosen fault/error handling model.

h. Can the current parameter values being used in the fault/error handling models be justified? How are those values derived from the system design?

i. How are other system or component fault tolerance features (e.g., memory scrubbing or error correcting codes) reflected in the model or model parameters?

j. How do the reliability prediction models map onto each other at various levels in the design hierarchy (e.g., system, module, device)? Especially, how are the lower level fault/error handling models mapped onto the higher level fault/error handling models?

k. What are the justifications for any assumptions not previously discussed? List all assumptions in the model or in the model parameters (i.e., failure rates and distributions, fault/error handling model parameters, any system feature ignored to make the model more tractable or not included since the modeler assumed those features did not have a significant impact on reliability).

l. What are the parameters that will require further evaluation and the possible methods for determining those values?

### 6.5.2 Modeling Analyses

The modeling analyses questions below address the issue of how the reliability prediction tools/methods are used as design aids. Specifically, these questions seek to determine if reliability prediction tools are correctly used as design aids to help select the best design from the set of potential designs and in refining the chosen design approach. The questions also try to establish how closely the reliability prediction effort is coordinated with the system design effort. Finally, the questions try to determine if any problems have been encountered in using the tools and what solutions are available:

a. What tradeoff or parametric analyses were performed? For example:
   1. architecture (various combinations of system designs and likely parameters)
   2. parametric studies of coverage
   3. parametric studies on the number of spares

4. parametric studies of fault/error handling model parameters
5. parametric studies gauging the impact of various permanent, transient, and intermittent failure rates on system reliability

b. What were the results of those analyses?
c. What feedback has been given to system designers as a result of the modeling analyses?
d. What future analyses will be performed, given the results of the modeling, and how might these analyses provide information to system designers?
e. What is the the current reliability assessment of the system design?
f. What, if any, difficulties have been encountered with the tools/ methods usage (e.g., the state space problem)? What solutions, if any, will be implemented?

### 6.5.3 Model Verification

The model verification questions below address the methods used to validate model parameters and how the data from those methods will be analyzed. Additionally, these questions ask how the results of the verification methods map onto the corresponding design model parameters. Finally, the questions try to establish how closely the reliability prediction function is coordinated with the coverage evaluation function:

a. What simulation or experimental methods will be used to determine model parameters and system reliability? How will the gathered data be analyzed for use in the reliability prediction tools/methods?
b. How do the simulation or experimental results map onto the reliability prediction model and model parameters?
c. How closely are the personnel that are performing reliability prediction working with the personnel performing coverage evaluation? How will the results of the coverage evaluation be used in the reliability prediction?
d. What techniques will be used to verify any remaining assumptions. If any assumptions are violated, how will this be reflected in the model and quantified?

## 6.6 INCLUDE APPROPRIATE COVERAGE VALUES

Reliability prediction tools should have the capability to allow appropriate coverage values to be included in an OBS model at any design level. Current tools only address coverage at the lowest level in the design hierarchy. Actual fault-tolerant designs typically have hierarchical treatment of errors, and not all errors are necessarily treated the same way at each level. Recovery is also typically hierarchical. If the fault tolerance mechanisms associated with the individual levels in the error handling hierarchy, together with their respective coverages, are not accurately included in the model, then the model will not provide a good approximation of the OBS, and will result in inaccurate reliability predictions.

## 6.7 INCLUDE CORRECT MODELING OF TRANSIENT FAULTS

Reliability prediction tools should have the capability to include and evaluate transient fault and error handling mechanisms at all levels in the design model hierarchy. Current tools address transients only at the lowest level of the design model hierarchy. When these low-level mechanisms cannot detect and recover from a transient, the current tools assume the system has failed. However, in actual designs, higher level mechanisms may be included to recover adequately from the fault's effect.

## 6.8 INCLUDE CORRECT MODELING OF SPARES THROUGHOUT THE DESIGN HIERARCHY

The tools' ability to accurately predict an OBS's reliability will be improved by the capability to accurately model warm and cold spares, when using time-varying failure rates. Currently, reliability prediction tools model all standby spares as hot, if time-varying failure rates are used. This limitation results in conservative reliability estimates. The magnitude of the inaccuracy cannot currently be predicted.

## 6.9 IMPROVED FEHMS

Reliability prediction tools require improved FEHMs. Tool developers have attempted to abstract the fault and error handling process into FEHMs that are generally applicable to any design. This approach to FEHMs is limited, since the approach provides little or no flexibility. A modeler must try to "fit" a design's fault and error handling into a FEHM whether the FEHM is an adequate abstraction of the fault and error handling process or not. Tools need to provide modelers the ability to create their own FEHMs, so the most accurate modeling of a design's fault and error handling may occur. Additionally, FEHMs should be extended to model a system's handling of near-coincident faults instead of treating their occurrence as a system failure.

# 7. SUMMARY

Three major error sources in the reliability prediction process were identified (i.e., model construction errors, tool limitations, and solution and representation errors). Model construction was pointed out to be the area where modelers can most effectively minimize errors because they can exercise significant control over this area, whereas they have little or no control over the remaining two error sources. Specific error sources in model construction were identified, (i.e., modeler assumptions and misunderstandings, incorrect parameter value determinations, and incorrect measurements).

A specific set of three tools was used to illustrate how to model correctly an example design in order to minimize the introduction of errors. By following the guidelines on model construction given in this report the variability in both the OBS models and the modeling results can be reduced, allowing evaluators to better understand and use the modeling results.

Modelers, designers, and system evaluators must use caution in inter-preting the predictions provided by the tools. Even if no errors occur in model construction or the tools, accurate model parameters are extremely difficult to obtain. As a result, a tool's results should not be considered an absolute measure of OBS reliability. That is, a reliability prediction containing a given number of digits does not imply a reliability prediction accurate to that number of digits. The results provide only relative measures of reliability allowing tradeoffs between different design approaches or refinements to a chosen design approach.

Guidelines on proper tool selection were given. The goal of tool selection is to choose tools that allow modelers to most accurately represent an actual design by a model. Modelers should avoid tools with inherent limitations that most limit the accuracy of the constructed model.

To further minimize errors in reliability prediction, recommendations to improve the specification of the reliability prediction process were presented. These recommendations included modifying MIL-STD 756B to include reliability prediction techniques that allow more accurate OBS models to be constructed. Recommendations were also given requiring system developers to verify model results, to present descriptions of the reliability prediction models themselves, and to predict the reliability of an example OBS. Also recommended was the use of a baseline set of questions that serves as a tool to help improve the reliability prediction process.

Four improvements can be made to reliability prediction tools that will allow modelers to construct more accurate OBS models. The improvements are:

1. Allow appropriate coverage values to be used at any design level in an OBS model.
2. Include the fault and error handling of transients at all levels in a design model hierarchy.
3. Allow the modeling of warm and cold spares, when time-varying fault arrival rates are used.
4. Allow modelers the ability to create their own FEHMs so more accurate modeling of fault and error handling occurs.

## BIBLIOGRAPHY


1. [Bavu 84a] S. Bavuso, J. Brunelle, and P. Peterson,"CARE III Hands-on Demonstration and Tutorial,"NASA Technical Memorandum 85811, May 1984.

2. [Bavu 84b] S. Bavuso, P. Peterson, and D. Rose,"CARE III Model Overview and User's Guide," NASA Technical Memorandum 85810, June 1984.

3. [CRAF 88] CRAFTS User Reference and Manual, Jan. 1988.

4. [Duga 84] J. Dugan, K. Trivedi, R. Geist, and V. Nicola, "Extended Stochastic Petri Nets: Applications and analysis, in *Performance '84*, E. Gelenbe, editor, pp. 507-519, Amsterdam, 1984, Elsevier Science Publishers B. V. (North-Holland).

5. [Duga 86] J. Dugan, K. Trivedi, M. Smotherman, and R. Geist, "The Hybrid Automated Reliability Predictor," *AIAA Journal of Guidance, Control, and Dynamics*, May-June 1986 pp. 319- 331.

6. [Duga 90] J. Dugan, S. Bavuso, and M. Boyd, "Fault Trees and Sequence Dependencies," *Proc. Annual Reliability and Maintainability Symposium*, Jan. 1990, pp. 286-293.

7. [HARP 86] "HARP: The Hybrid Automated Reliability Predictor Introduction and User's Guide," NASA Langley Research Center, September 1986.

8. [HARP 89] "HARP: The Hybrid Automated Reliability Predictor Introduction and User's Guide HARP Version 6," NASA Langley Research Center, April 1989.

9. [Maka 82] S. Makam and A. Avizienis, "ARIES 81: A Reliability and Life-Cycle Evaluation Tool for Fault-Tolerant Systems," *Proceedings IEEE 12th Fault-Tolerant Computing Symposium*, June 1982, pp. 267-274.

10. [MIL 81] MIL-STD 756B, "Reliability Modeling and Prediction," Nov. 1981.

11. [MIL 86] MIL-HDBK 217E, "Reliability Prediction of Electronic Equipment," Rome Air Development Center, Oct. 1986.

12. [Ng 76] Y. Ng and A. Avizienis, "A Model For Transient and Permanent Fault Recovery In Closed Fault-Tolerant Systems,"*Proceedings IEEE 6th Fault-Tolerant Computing Symposium*, June 1976, pp. 182-188.

13. [Ng 80] Y. Ng and A. Avizienis, "A Unified Reliability Model for Fault-Tolerant Computers," *IEEE Transactions on Computers*, Vol. C-29, No. 11, Nov. 1980, pp. 1002-1011.

14. [Triv 84] K. Trivedi, "Reliability Evaluation for Fault-Tolerant Systems," in *Mathematical Computer Performance and Reliability*, ed. G. Iazeolla, P. Courtois, and A. Hordijk, pp. 403-414, North-Holland, Amsterdam, 1984.